Toward Securing Links and Large-Scale Networks of Resource-Limited Devices

A Thesis
Presented to
The Academic Faculty

by

Farshid Delgosha

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2007

Toward Securing Links and Large-Scale Networks of Resource-Limited Devices

Approved by:

Professor Faramarz Fekri
Advisor, Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Steven W. McLaughlin
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Raghupathy Sivakumar
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Chuanyi Ji
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Alexandra O. Boldyreva
Department of Computer Science
Georgia Institute of Technology

Date Approved: 27 July 2007

To the memory of my mother and

to my father and my brother

whom I love with all my heart.

ACKNOWLEDGEMENTS

I would like to express my appreciation to my advisor, Prof. Faramarz Fekri, whose knowledge, wisdom, experience, and caring paved the way for success in my academic career. His support will always be remembered.

The support and love that I have received from my family have been enormous. I would like to highly appreciate my father Mohammad and my brother Payam for being so kind.

I also take the opportunity to thank my colleagues and friends, specially Dr. Babak Firoozbakhsh, Dr. Majid Fozunbal, Dr. Pejman Monajemi, Ms. Gail Palmer, and Mr. Erman Ayday, whose support and caring have always enlightened my way during the years at Georgia Tech. I am very grateful for their friendships.

Finally, I wish to highly thank Ms. Shirley McKelheer for her continuous love, support, and encouragements that have kept my heart warm for so many years. She will always remain in my memory and my heart.

TABLE OF CONTENTS

Part I

Bivariate PU Filter Banks over Fields of Characteristic Two

Part II

Multivariate Cryptography

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

Part I

DTWT	discrete-time wavelet transform
FIR	finite impulse response
IIR	infinite impulse response
LOT	lapped orthogonal transform
PR	perfect reconstruction
PU	paraunitary
TDFBC	two-dimensional filter-bank code

Part II

AES	advanced encryption standard
CFB	cipher feedback
DES	data encryption standard
ECC	elliptic curve cryptography
ECDSA	elliptic curve digital signature algorithm
FSM	finite-state machine
FXL	fixing and extended relinearization
HFE	hidden-field equations
LDPC	low-density parity-check
LFSR	linear feedback shift register
ML	maximum likelihood
NC	nonlinear combiner
NF	nonlinear filter
OFB	output feedback
OWF	one-way function
PAC	paraunitary asymmetric cryptosystem
PDSS	paraunitary digital-signature scheme
SSC	self-synchronizing stream cipher
TOWF	trapdoor one-way function
TRMC	tractable rational-map cryptosystem
TTM	tame transformation method
TTS	tame transformation signature

WSSC	wavelet self-synchronizing stream cipher
XL	extended relinearization

Part III

CH	cluster head
Du	Du scheme
EG	Eschenauer-Gligor scheme
GB	grid-based scheme
HB	hypercube-based scheme
IHA	interleaved hop-by-hop authentication
KPS	key pre-distribution scheme
LBRS	location-based resilient security
LEDS	location-aware end-to-end data security
LA-GB	location-aware grid-based scheme
LA-MKPS	location-aware multivariate key pre-distribution scheme
LNCS	location-aware network-coding security
MAC	message authentication code
MDS	maximum distance separable
MKPS	multivariate key pre-distribution scheme
qComp	q-composite scheme
RPB	random polynomial-based scheme
SEF	statistical en-route filtering
WSN	wireless sensor network

SUMMARY

This thesis is concerned with both link and network security. It explores applications of wavelets and paraunitary (PU) matrices over finite fields in cryptography to provide link and network security for resource-limited devices. It also exploits algebraic techniques to secure wireless sensor networks. Algebraic natures of these tools allow not only the designation of highly efficient cryptographic algorithms but also their security analysis in a systematic manner. These advantages and features can be compared with the traditional methods that are mostly ad hoc.

The primary focus of this thesis is cryptography and network security. However, since multivariate PU matrices are one of the main ingredients in our designs, we study their factorization in Part I. In a factorization method, one seeks a small set of fully-parameterized building blocks that their multiplication generates PU matrices. By taking advantage of the lifting method, we propose a multi-level technique that converts the problem of factoring bivariate PU matrices into their univariate counterpart. Precisely, our contributions in Part I are briefly as follows.

1. We propose, for the first time, fully-parameterized 2×2 bivariate PU building blocks along with a two-level factorization algorithm.

2. Using the newly proposed building blocks, we are able to generate a subclass of bivariate PU matrices that is larger than the subclass generated using other methods.

3. The proposed bivariate PU building blocks have some practical applications. For example, they can be used to construct two-dimensional error-correcting encoders and syndrome generators.

The design of multivariate cryptographic primitives is the main goal in Part II. In this part, first we use finite-field wavelets as sequence transformers to design the wavelet

self-synchronizing stream cipher (WSSC). Although the new design mimics the traditional canonical model, there are some fundamental differences, which are discussed. For example, the secret key specifies the wavelet coefficients instead of determining the initial state of a shift register. The security analysis of the proposed stream cipher indicates that it is invulnerable to methods such as the correlation attack that threatens the security of many designs based on linear feedback shift registers (LFSRs). Our contributions and innovations in the stream-cipher design are summarized below.

1. None of the attacks developed for the state of the art stream ciphers is applicable on the WSSC since it is not based on LFSRs.

2. By taking advantage of the relationship between wavelets and PU matrices, we provide an efficient method to setup the WSSC. Short setup time is beneficial when stream ciphers are implemented on resource-limited devices such as cell phones or PDAs.

3. We provide estimates of the circuit complexities of the WSSC and AES in the one-bit cipher feedback mode. Our results reveal that the WSSC has less circuit complexity.

In the remainder of Part II, we propose a new framework for the design of multivariate public-key cryptosystems. The new approach is based on the observation that the columns of every PU matrix serve as basis vectors for the module of polynomial vectors. In Part II, we explain how to use this observation to design a public-key cryptosystem and a digital signature scheme. Algebraic properties of PU matrices allow establishing a connection between the difficulty of breaking the proposed schemes and the difficulty of solving systems of multivariate polynomial equations as the underlying mathematical problem. In addition to our results on the computational security of the proposed schemes, we have studied several algebraic attacks. Our results indicate that the proposed schemes resist all studied attacks with proper choices of design parameters. Briefly, the following have been achieved in Part II.

1. We propose a completely novel method to design multivariate cryptosystems using PU matrices. In this new method, we establish a link between breaking any system

designed based on our approach and a long standing open mathematical problem. No such mathematical analysis exists in the previous work on multivariate cryptography.

2. Using our proposed new approach, we provide a practical public-key cryptosystem and a digital signature scheme. Our complexity estimates reveal that our systems either outperform previous designs or have comparable complexities.

Motivated by the wealth of algebraic methods, we investigate their applications in the key pre-distribution and data authentication for wireless sensor networks in Part III. In this part, we first propose a multivariate key pre-distribution scheme (MKPS) for wireless sensor networks. In this method, sensor nodes are loaded with shares of multivariate polynomials prior to the node deployment. After the network is deployed, nodes use their pre-loaded information to establish pairwise keys among their one-hop neighbors. We show that this method outperforms all previous techniques in terms of the resiliency against node capture. In addition, we propose a location-aware version of the MKPS. In the following, we provide a summary of our contributions in the key pre-distribution for sensor networks.

1. The MKPS significantly improves the resiliency of the network against the node capture without any payoffs such as increase in the node memory or communication range.

2. We have proposed an algorithm to optimize the MKPS with respect to (1) the probability of secure link establishment (network connectivity) and (2) the probability of link compromise (network resiliency). Considering both these criteria as design parameters allows the network designer to increase the network lifetime while maintaining a desirable level of resiliency. None of the previous schemes offers such robustness in design.

3. By taking advantage of information about the approximate locations of nodes on the field, we propose the location-aware multivariate key pre-distribution scheme (LA-MKPS) in Chapter 7. This scheme provides perfect network connectivity while significantly improving the network resiliency compared to previous location-aware schemes.

In Part III, in addition to data confidentiality, we study three other security services that are: (1) entity authenticity, (2) data integrity, and (3) data availability. These services are used to protect a network against eavesdropping, false data injection, data drop, and noise injection. In Chapter 8, we propose location-aware network-coding security (LNCS) that offers all the aforementioned security services to wireless sensor networks. In this scheme, data is forwarded toward the sink via multiple paths, and it is collaboratively authenticated by multiple nodes along the path. Our contributions and improvements over previous schemes are as follows.

1. To our knowledge, LNCS is the first scheme that practically and feasibly provides all the three aforementioned security services.

2. For the first time, the LNCS employs random network coding that intrinsically provides data availability. This technique generates redundant information to facilitate the recovery of packets erased by the channel or dropped by malicious nodes. This kind of coding significantly improves data availability compared to all other schemes.

3. In contrary to previous schemes, our proposed scheme do not require a trustworthy cluster head (CH) that is responsible for generating the report and forwarding it to the next cell. We emphasize that the existence of a trustworthy CH cannot be guaranteed, and a malicious CH completely breaks down the security of the protocol.

CHAPTER 1

Introduction and Background Review

In our modern day in which massive amounts of information are communicated throughout the globe, the need for efficient and reliable information processing devices has become a necessity more than ever. Used on a daily basis, PDAs, laptops, cell phones, USB flash drives, and smart cards are a few portable devices that store or process sensitive information such as governmental, military, commercial, and personal. In most cases, data confidentiality and authenticity are among the crucial security services that must be available to users [114, 140, 100, 163]. The need for security arises from the fact that portable devices communicate with the outside world through the wireless channel. Hence, eavesdropping or mounting active attacks could be very easy for an adversary attempting to gain access to the private information or simply to disrupt the flow of information. In addition to the obvious necessity for secure portable connectivity, the secure storage of data is crucial as well to protect compromising highly sensitive information in case a portable storage device (such as a USB flash drive) is lost or stolen [196].

In spite of the considerable progress in the science of information security since World War II, there is still a constant need for efficient cryptographic algorithms. Efficiency is particularly a primary factor when an algorithm is implemented on resource-limited devices where energy consumption is the bottleneck determining the lifetime of the device. Among the cryptographic primitives, public-key cryptosystems are usually the most complex but the most useful ones [119]. Therefore, the design of highly efficient public-key primitives has been the topic of research for many years. In addition, it would be desirable to have practical cryptosystems based on problems other than the handful of assumptions currently in use

such as RSA and discrete logarithm. We might be in a safer state against possibilities such as the emergence of an efficient algorithm for factoring or computing discrete logarithms.

Motivated by these observations, many cryptographers and mathematicians have attempted to investigate the applications of algebraic techniques in the design of public-key primitives in a multivariate setting. Multivariate cryptography, a new branch of cryptology, encompasses cryptosystems describable by a set of well-defined algebraic equations over a small-size algebraic structure such as a finite field or a ring. The security of multivariate cryptosystems is based on the intractability of solving systems of multivariate polynomial equations over finite fields or rings [1, 13]. To help visualize the standing of multivariate cryptosystems with respect to the RSA, a well-studied cryptosystem, we provide a logical interconnection. RSA is described by a single monomial over a ring of integers defined by two very large primes. However, multivariate cryptosystems are usually specified by several multivariate polynomials over a small finite field. Therefore, one expects the latter to perform much faster than the RSA since all arithmetics are carried out over a small-size algebraic structure. The improvement in speed comes with the price of increase in the size of the memory required to store the description of the system, i.e., the coefficients and exponents of the multivariate polynomials. Nevertheless, the energy consumption, among all other factors, usually dominates in determining the lifetime of most portable devices. Hence, multivariate cryptosystems are suitable for applications in resource-limited environments.

The attraction of portable devices comes from the fact that they are compact yet can carry out many tasks on their own. Hence, the idea of constructing a network of very small and portable devices that are able to sense some attribute of the environment seems very attractive. As a matter of fact, such distributed networks of wireless sensors have received a lot of attention from the research society [5]. The generic definition of wireless sensor networks (WSNs) serves as a descriptive template for many real-world problems such as surveillance, energy management, medical monitoring, etc. [9]. A typical WSN may consist of hundreds to several thousands of sensor nodes that are low cost and battery powered and have limited computation power and memory. Sensor nodes are either randomly or manually

scattered in a field. They form an unattended wireless network that collects information about the field such as temperature, illumination, motion, some chemical material, etc. The collected data is partially aggregated and forwarded to a central processing unit, called the sink, that is responsible for interpreting the data and taking appropriate actions (e.g., sending personnel for precise measurements).

Security is a critical issue when WSNs are deployed in a hostile environment. An adversary can monitor the traffic, capture or impersonate sensor nodes, and provide misleading information. An essential security primitive, which is a building block for many security services, is pairwise key establishment, referred to as key establishment. Although public-key cryptography provides a complete solution to this problem, constraints in the node processing capabilities and also limitations in the networking features demand new solutions in WSNs. Data authenticity is another security concern that is automatically addressed in a public-key infrastructure using digital signature schemes. Nevertheless, the complexity of existing signature schemes renders them inapplicable in WSNs.

The design of highly efficient cryptographic algorithms using algebraic techniques is the main goal of this thesis. Precisely, the objectives of this thesis are fourfold:

1. devising a technique to factorize bivariate paraunitary (PU) matrices over finite fields into the product of fully-parameterized building blocks,
2. developing a new self-synchronizing stream cipher system based on finite-field wavelets,
3. proposing an algebraic framework for designing new multivariate public-key cryptosystems and signature schemes using multivariate PU matrices over finite fields, and
4. developing algebraic multivariate key pre-distribution and authentication schemes for WSNs.

In the following, we summarize previous works and our contributions in each of these topics.

Bivariate PU matrices are a subclass of invertible polynomial matrices, i.e., matrices which their entries are polynomials. Algebraic properties of PU matrices make them attractive tools in many mathematical and engineering problems. These properties include: (1) every PU matrix is invertible by its definition and (2) the inverse of every PU matrix, which is a polynomial matrix itself, can be obtained with no computation. The usefulness of PU matrices is highlighted when there exist methods to efficiently generate them. Fortunately, this is the case for *univariate* PU matrices, i.e., there are a few fully-parameterized building blocks that their multiplications generate all univariate PU matrices. This fact has significant practical importance since: (1) every building block is determined by only a few parameters (an advantage for software implementation) and (2) any multiplication of these factors is translated to a cascade of parameterized systems (important for hardware implementation).

Any PU matrix is specified by a set of multivariate polynomial equations. Therefore, one may generate such matrices by simply solving their corresponding systems of equations. (It is also possible to add new equations to impose extra properties on the resulting PU matrix.) Two main disadvantages of this method are: (1) it is inefficient in the average since there is no known polynomial-time algorithm to solve systems of multivariate polynomial equations and (2) even if a PU matrix is generated using this method, it is not clear how to implement it specifically in hardware. Fortunately, there exists an efficient factorization method that allows representing any univariate PU matrix as a product of the building blocks. As a summary, two problems of interest are:

1. devising an efficient and systematic method to generate all PU matrices and
2. providing an efficient algorithm for representing any given PU matrix in an implementable format.

As explained before, there are efficient algorithms for both problems in the case of univariate PU matrices.

Considering the interesting features of univariate PU building blocks and the efficiency of their factorization method, there have been attempts to seek such building blocks and factorization algorithm for bivariate (and possibly multivariate) PU matrices. Unfortunately, the extension from the univariate to the bivariate case is not trivial. One of the naive methods, suggested in the literature, is using univariate building blocks in different variables. However, this method neither generates nor allows factoring all bivariate PU matrices. Another incomplete method proposed by some researchers is using the Kronecker product of two univariate PU matrices each in one of the variables. As an alternative method, it has been suggested to use the Gröbner bases (a classical and universal method for solving systems of multivariate polynomial equations) to generate multivariate PU matrices. As we will explain later, the complexity of this method is double exponential in the number of variables. In addition, it does not provide a systematic way to implement a multivariate PU system.

Since none of the previous methods has been successful, in Chapter 3, we propose a new method for generating and factorizing 2×2 bivariate PU matrices over fields of characteristic two. Our contributions and improvements over previous methods are as follows.

1. We propose fully-parameterized bivariate PU building blocks along with a two-level factorization algorithm. This method always allows the first level of factorization that is over a ring of polynomials in one of the variables. The possibility of the second level of factorization depends on the factors obtained in the first level. Although there are bivariate PU matrices that cannot be factored using this method, the ones that are factorable are represented as a product of fully-parameterized building blocks. The importance of such representation in terms of implementation was explained before.

2. Using the newly proposed bivariate building blocks, we are able to generate a subclass of bivariate PU matrices that is larger than the subclass generated by multiplying univariate building blocks and other methods. Therefore, the new method is a step forward although not complete.

3. The proposed bivariate PU building blocks have some practical applications. For example, they can be used to construct two-dimensional error-correcting encoders and syndrome generators. Previously proposed methods does not have such useful properties.

This is a new trend in cryptography in which algebraic methods over small-size fields or rings are used to construct highly efficient public-key cryptographic algorithms. Two main motivations for studying such systems are:

1. Existing public-key schemes are computationally too complex for resource-limited devices such as PDAs, smart cards, wireless sensor nodes, etc.
2. We might be in a safer state against possibilities such as the emergence of an efficient algorithm for factoring or computing discrete logarithms.

The security of currently in use public-key schemes is based on the difficulty of either factoring large composite integers or computing discrete logarithm over large cyclic groups. Cryptographic methods based on these two problems are complex since the underlying algebraic structure is very large in size. To significantly improve efficiency, multivariate cryptosystems base their security on the difficulty of solving systems of multivariate polynomial equations over small finite fields. There is no known polynomial-time algorithm to solve this problem. The classical method that completely solves this problem *in theory* is Gröbner bases that its complexity is, in the average, double exponential in the number of variables (unknowns) that we denote by n. Besides the number of variables, another important factor in the complexity of solving systems of polynomial equations is the number of equations m. If m is significantly larger than n (i.e., the system of equations is over-defined), the system can be efficiently solved using methods such as linearization and its variations such as extended relinearization (XL) and fixing and extended relinearization (FXL). These methods are much less complex than the Gröbner bases method only when m is in the order

of n^2. Therefore, we focus our attention to systems of equations in which either $m \approx n$ or at most m is linearly related to n.

Solving systems of homogenous quadratic equations over small finite fields is also considered as a hard problem. As a subproblem of solving systems of general polynomial equations, this has been the underlying hard problem for almost all related work in multivariate cryptography so far. The interest in quadratic homogenous polynomials comes from the fact that their special structure limits the number of monomials. Hence, the amount of storage memory required to store the coefficients of the polynomials is limited. Public-key cryptosystems designed based on this approach include C^* algorithm, hidden-field equations (HFE), big dragon, and tame transformation method (TTM). Examples of signature schemes design based on this approach are tame transformation signature (TTS), QUARTZ, FLASH, and SFLASH. Unfortunately, all these schemes have been shown to be insecure due to the existence of unexpected algebraic relations.

Considering the insecurity of previous schemes in multivariate cryptography, we study these systems and seek a systematic method for their designation in Chapter 6. Our main contributions in this chapter are as follows.

1. We propose a completely novel method to design multivariate cryptosystems using PU matrices. In this new method, we establish a link between breaking any system designed based on our approach and a long standing open mathematical problem. In other words, we prove that if the mathematical problem has a positive answer, then efficiently breaking any instance of our approach is equivalent to finding an efficient algorithm for solving systems of multivariate polynomial equations. We note that the aforementioned open mathematical problem is not a computational problem. It has either a positive or a negative answer.

   The ability of establishing such link between the difficulty of breaking a cryptosystem and a non-computational mathematical problem stems from interesting algebraic properties of PU matrices. We emphasize that none of the previous schemes has been able to provide such evidence on the security of their method. They just simply make claims

about the difficulty of breaking their systems in relation to solving a hard mathematical problem. Therefore, our proposed approach is one step forward toward the design of provably secure multivariate cryptosystems.

2. Using our proposed new approach, we provide a paraunitary asymmetric cryptosystem (PAC) and a paraunitary digital-signature scheme (PDSS). In addition, we estimate complexities of different algorithms associated with these schemes (such as encryption and signature generation). Comparing with complexities of the same algorithms in HFE-like systems, we deduce that our systems either outperform previous designs or have comparable complexities while maintaining a desirable security level.

Considering the attractive features of public-key multivariate cryptosystems (such as efficiency), we investigate the possibility of employing such techniques in the design of self-synchronizing stream ciphers in Chapter 5. Most of the related work in stream cipher design is based on linear feedback shift registers (LFSRs) and nonlinear Boolean functions. Nevertheless, many powerful attacks have been developed for such systems such as correlation attack. An alternative design for self-synchronizing stream ciphers is using any block cipher (such as AES) in the cipher feedback mode that effectively converts it to a self-synchronizing stream cipher. However, a stream cipher with a dedicated design is expected to perform faster. In Chapter 5, we employ finite-field wavelets as algebraic sequence-transformers to propose a wavelet self-synchronizing stream cipher (WSSC). Our interest in wavelet transformations originates from their rich mathematical foundation. Our new approach to the design of stream ciphers can be considered as a novel design methodology. The WSSC is categorized a multivariate cryptosystem since it can be completely described using polynomial equations over a non-binary field. Attractive features of this new design and our contributions are as follows.

1. None of the attacks developed for the state of the art stream ciphers is applicable on the WSSC since it is not based on LFSRs. In addition, our studies show that none of the existing algebraic attacks also poses a threat on the WSSC.

2. By taking advantage of the relationship between wavelets and PU matrices, we provide an efficient method to setup the WSSC. The efficiency of the setup algorithm comes from the fact that PU matrices can be efficiently generated by multiplying fully-parameterized PU building blocks. Short setup time is beneficial when stream ciphers are implemented on resource-limited devices such as cell phones or PDAs.

3. We provide estimates of the circuit complexities of the WSSC and AES in the one-bit cipher feedback mode. Our results reveal that the WSSC has less circuit complexity. This observation, although not a definitive proof of superiority in speed, provides some insight on the potential efficiency of the WSSC after implementation. The improvement in speed comes with the increase in the amount of required storage memory.

Improvements in micro-electro-mechanical systems have paved the way for the design and cost-efficient massive production of wireless sensor nodes. A sensor node is a battery powered device equipped with a processor (with low computational power), a sensing unit, and a wireless communication unit. The objective is to construct a wireless network of sensor nodes that are randomly scattered in a field. Triggered by an event in the field, the sensor nodes compile a report of their own sensor readings and send it back to a central unit, called the sink, in a hop-by-hop manner. The sink after collecting the sensor readings makes a decision such as sending personnel for more accurate readings.

Security is a critical issue when sensor networks are deployed in a hostile environment. An adversary can monitor the traffic, capture or impersonate sensor nodes, and provide misleading information. An essential security primitive, which is a building block for many other security services, is pairwise key establishment referred to as *key establishment*. Public-key cryptography provides a complete solution in traditional networks. However, public-key algorithms are too complex for sensor nodes. Moreover, the lack of infrastructure makes it difficult to provide certificates for public keys. Key pre-distribution is a feasible solu-

tion to the key establishment problem in sensor networks. In this approach, sensor nodes are loaded with some keying material prior to their deployment in the field. After being scattered in the field, nodes try to establish pairwise keys with their neighbors using the preloaded information.

The previous work in key pre-distribution schemes (KPSs) can be generally categorized as random and deterministic. In the former, keying materials are randomly selected from a large pool of keys and stored in nodes. After the network deployment, neighbor nodes with common keys can establish secure links. We emphasize that two nodes can establish a link if and only if (1) they are within the communication range of each other and (2) they are able to establish a common key. Therefore, to improve the probability of network connectivity (which is dependent on the probability of node connectivity using the random graph model), the number of keys stored in every node must be increased. By this increase, the number of keys revealed to an adversary upon the physical capture of any node increases. Thus, as an alternative solution to the key pre-distribution problem, deterministic KPSs have been proposed in which the keying materials for every node are deterministically selected based on an ID uniquely assigned to that node. Combining this approach with a threshold key establishment scheme, some deterministic KPSs have been developed that significantly improve the resiliency of the network against the node capture. Polynomial-based threshold key establishment is one of the mostly used schemes in the literature.

One approach taken in the literature to improve the network connectivity while maintaining the resiliency at the level provided by deterministic schemes is to take advantage of the deployment knowledge when such information is available. For this purpose, one of the proposed methods is to assign a probability distribution to the actual placement of every node in the field. Although attractive in theory, this method is very difficult to implement in practice. Another technique studied in the literature is to divide the terrain into non-overlapping cells and uniformly at random scatter nodes inside every cell. With this technique, although every node is unaware of its exact location on the field, it can determine the center of its residing cell (e.g., by using a localization scheme). This technique is more

practical than the previous one.

In Chapter 7, we propose two new multivariate KPSs that are multivariate key pre-distribution scheme (MKPS) for random node deployment and location-aware multivariate key pre-distribution scheme (LA-MKPS). In these schemes, node IDs are selected from points on a hypercube in a multidimensional space. Moreover, symmetric multivariate polynomials are used for the first time in such schemes. Our contributions and improvements over previous schemes are as follows.

1. The MKPS significantly improves the resiliency of the network against the node capture without any payoffs such as increase in the node memory or communication range. This improvement is related to the fact that any existing link between two nodes is secured with a fixed number of common keys. Therefore, an adversary attempting to compromise the secure link between any two un-capture nodes has to compromise all the common keys.

2. We propose an algorithm to search for the optimum dimension for the hypercube employed in the scheme. The optimality criteria are: (1) the probability of secure link establishment (network connectivity) and (2) the probability of link compromise (network resiliency). The MKPS is the first scheme that employs both these criteria as design parameters. Using the random graph model for the network, the optimization algorithm searches only over dimensions for which the network has a giant component (a large connected subnetwork). The philosophy behind this choice is that the existence of a giant component suffices for a network to carry out its normal functions. All the previous work attempt to design a completely connected network, which may require an unnecessarily high probability of the link establishment.

   In the MKPS, taking into account the network connectivity as an optimization criterion provides another advantage over previous schemes as follows. If the network designer somehow knows that the percentage of captured nodes will not increase beyond some threshold (e.g., because it will become too costly for an adversary), it is an over-

design to provide resiliency to the network beyond that threshold. By decreasing the network resiliency, one can improve the network connectivity and, as a result, decrease the communication range. This provides a significant saving in energy (and network lifetime) considering that the energy consumption for the radio communication is directly proportional to a fixed power of the communication radius.

3. In the LA-MKPS, the field is divided into non-overlapping cells, and nodes are randomly scattered in every cell. Two layers of MKPS are employed to secure the inter- and the intra-cell communications. Comparing to previous location-aware schemes, the LA-MKPS significantly improves the network resilience against the node capture.

In addition to data confidentiality, three other security services necessary in WSNs are: (1) entity authenticity, (2) data integrity, and (3) data availability. These services are used to protect a network against the following threats. (1) *Eavesdropping:* By listening to the radio channel, the adversary tries to obtain meaningful information. (2) *False Data Injection:* In this attack, an insider node attempts to cause false alarms or to consume the energy of the forwarding sensors by injecting false data. (3) *Data Drop:* An insider node drops a legitimate report on the forwarding path toward the sink. (4) *Noise Injection:* Legitimate reports are modified by injecting noise. Thus, the sink is unable to recover the original message.

To address the need for entity authenticity and data integrity, a few schemes have been proposed in the literature including interleaved hop-by-hop authentication (IHA), statistical en-route filtering (SEF), and location-based resilient security (LBRS) that employ some message authentication code (MAC) mechanism. In all these schemes, the report generated at the center of an stimulus is forwarded toward the sink in a hop-by-hop fashion. In every hop, a single node on the forwarding path transmits the data to the next node on the path. Therefore, a malicious node on the path can simply drop the data. Hence, none of these schemes provides data availability. The only existing scheme (to our knowledge) that attempts to provide data availability in addition to other security services is the location-

aware end-to-end data security (LEDS). The main drawback of this technique is that it requires overhearing nodes and a voting mechanism to assure data availability with some probability. Although justifiable in theory, there does not seem to exist a practical and feasible method to implement this technique.

Considering the shortcomings of previous works in proposing a practical solution that provides all the aforementioned security services, we propose a novel authentication scheme called location-aware network-coding security (LNCS) in Chapter 8. Our contributions and innovations in the design of this scheme are as follows.

1. In the proposed scheme, data is forwarded toward the sink via multiple paths, and it is collaboratively authenticated by multiple nodes along the path. This scheme performs data authentication without overhearing nodes and voting systems. In the LNCS, nodes generate the authentication information using a hash tree on the encoded data. The existence of highly efficient algorithms to construct hash trees and to use the information generated by them to authenticate data implies that the proposed LNCS is an efficient algorithm.

2. For the first time, the LNCS employs random network coding that intrinsically provides data availability. This technique generates redundant information to facilitate the recovery of packets erased by the channel or dropped by malicious nodes. This kind of coding significantly improves data availability compared to all other schemes.

3. In contrary to previous schemes, our proposed scheme do not require a trustworthy cluster head (CH) that is responsible for generating the report and forwarding it to the next cell. We emphasize that the existence of a trustworthy CH cannot be guaranteed, and a malicious CH completely breaks down the security of the protocol.

1.1 Notation

In this section, mathematical notations used throughout the thesis are introduced. Some notations are specific to a particular part or chapter of the thesis that are explained at the

beginning of that part or chapter. We have tried to employ standard notation. Therefore, one may skip this section and use it as a reference whenever needed. In the following, the notations are categorized based on their mathematical relevance.

1.1.1 Set Notation

Frequently-used set notations are summarized in Table 1.1. In many cases, it is required to refer to only a subset of a set $\mathcal{A}$ that satisfies a logical operator $\ell : \mathcal{A} \to \{0, 1\}$. The notation $\mathcal{A}_\ell$ is used for this purpose, i.e., $\mathcal{A}_\ell := \{ x \in \mathcal{A} : \ell(x) = 1 \}$. For example, $\mathbb{Z}_{\geq 0} = \{0, 1, 2, \dots\}$. The symbol $\mathbb{F}$ is used for Galois fields whenever the size of the field is clear from the context or the statements hold for all or some specified Galois fields. We define special notations for the following two subsets of integers since they are frequently used.

$$[\, n \,] := \{ x \in \mathbb{N} : x \leq n \} \qquad \forall n \in \mathbb{N} \tag{1.1a}$$

$$(\, n \,) := \{ x \in \mathbb{Z}_{\geq 0} : x \leq n - 1 \} \qquad \forall n \in \mathbb{N} \tag{1.1b}$$

For example, $[\, 3 \,] = \{1, 2, 3\}$ and $(\, 4 \,) = \{0, 1, 2, 3\}$.

Sets defined in Table 1.1 are usually used to construct other algebraic structures. As-

Table 1.1: Frequently-used set notations.

Symbol	*Meaning*
$\mathbb{N}$	Natural numbers $\{1, 2, 3, \dots\}$
$\mathbb{Z}$	Integers $\{0, \pm 1, \pm 2, \pm 3, \dots\}$
$\mathbb{R}$	Real numbers
$\mathbb{C}$	Complex numbers
$\mathbb{F}_q$	Galois field $\mathsf{GF}(q)$ with addition operator $+$ and multiplication operator $\cdot$
$\mathbb{F}$	A general notation for all Galois fields
$\mathfrak{S}_n$	Symmetric group of all permutations of n symbols

suming that $\mathcal{A}$ represents an arbitrary set, notations employed for such constructions are summarized in Table 1.2. In this table, the notation $\mathcal{A}^\times$ denotes the multiplicative subgroup of $\mathcal{A}$. Thus, if $\mathcal{A}$ is a field, then $\mathcal{A}^\times = \mathcal{A} \setminus \{ 0 \}$. The set $\mathcal{A}^*$, referred to as *signal space* or *sequence space*, consists of finite sequences, such as x, each of which is a mapping $x : \mathbb{Z}_{\geq 0} \to \mathcal{A}$. The set $\mathbb{Z}_{\geq 0}$ is referred to as the set of *indices* for sequences. The n-th element of x is denoted by either $x(n)$ or x_n. A sequence is also referred to as a *signal*. Let $n_f \in \mathbb{Z}_{\geq 0}$ and $n_l \in \mathbb{Z}_{\geq 0}$ be the indices of the first and the last nonzero elements of a sequence x, respectively. The set $\{ n \in \mathbb{Z}_{\geq 0} : n_f \leq n \leq n_l \}$ is called the *support* of x. To study a sequence, it suffices to know its values for all indices in its support. The set $\mathcal{A}^*$

Table 1.2: Sets constructed from an arbitrary set $\mathcal{A}$.

Symbol	*Meaning*
$\lvert\mathcal{A}\rvert$	Cardinality of $\mathcal{A}$
$\mathcal{A}^n$	Column vectors of length n with entries from $\mathcal{A}$
$\mathcal{A}^\times$	Largest multiplicative group contained in $\mathcal{A}$
$\mathcal{A}^*$	Finite sequences (or streams), including the empty sequence, constructed from the alphabet $\mathcal{A}$
$\mathcal{A}[x]$	Polynomials in nonnegative powers of x with coefficients from $\mathcal{A}$
$\mathcal{A}[x^{-1}]$	Polynomials in non-positive powers of x with coefficients from $\mathcal{A}$
$\mathcal{A}[x^{\pm 1}]$	Laurent polynomials in x with coefficients from $\mathcal{A}$
$\mathcal{A}(x)$	Rational functions in nonnegative powers of x with coefficients from $\mathcal{A}$
$\mathsf{M}_{n,k}(\mathcal{A})$	$n \times k$ matrices with entries from $\mathcal{A}$
$\mathsf{U}_n(\mathcal{A})$	$n \times n$ unitary matrices over $\mathcal{A}$
$\mathsf{PU}_n(\mathcal{A})$	$n \times n$ paraunitary matrices over $\mathcal{A}$
$\mathsf{GL}_n(\mathcal{A})$	General linear group over $\mathcal{A}$, i.e., the set of all $n \times n$ invertible matrices with entries in $\mathcal{A}$

becomes a vector space when $\mathcal{A}$ is a finite field. In this case, the vector addition and scalar multiplication operations are defined as follows. The addition of every two vectors $x, y \in \mathcal{A}^*$ is a new vector $x + y = z \in \mathcal{A}^*$ such that $z_n := x_n + y_n$ for all $n \in \mathbb{Z}_{\geq 0}$. Moreover, for all $a \in \mathcal{A}$, we have $a \cdot x = z \in \mathcal{A}^*$, where $z_n := ax_n$ for all $n \in \mathbb{Z}_{\geq 0}$. For any $\ell \in \mathbb{N}$, the set of length-ℓ vectors that take their elements from $\mathcal{A}$ can be considered a vector subspace of $\mathcal{A}^*$ consisting of sequences whose support is a subset of (ℓ).

The notation $\mathcal{A}[x]$ represents all polynomials in the nonnegative powers of the indeterminate x with coefficients from $\mathcal{A}$. Every polynomial $a \in \mathcal{A}[x]$ is of the form $a(x) = a_0 + a_1 x + \cdots + a_L x^L$ for some $L \in \mathbb{Z}_{\geq 0}$ in which $a_L \neq 0$. The highest exponent L, with the corresponding term having nonzero coefficient, is referred to as the *order* or *degree* of a with respect to x and is denoted by $\mathsf{Ord}_x a$ or $\mathsf{Deg}_x a$, respectively. Similarly, the notation $\mathcal{A}[x^{-1}]$ represents polynomials of the form $a(x) = a_0 + a_1 x^{-1} + \cdots + a_L x^{-L}$ for some $L \in \mathbb{Z}_{\geq 0}$ in which $a_0, \ldots, a_L \in \mathcal{A}$. Polynomials with both positive and negative exponents are called *Laurent* polynomials.

Definition 1.1.1 (Laurent Polynomial). *Let $\mathcal{A}$ be an arbitrary set. A Laurent polynomial over $\mathcal{A}$ is a polynomial $a(x)$ of the form*

$$a(x) = a_{-M} x^{-M} + a_{-M+1} x^{-M+1} + \cdots + a_{-1} x^{-1} + a_0 + a_1 x + \cdots + a_{L-1} x^{L-1} + a_L x^L$$

in which $L, M \in \mathbb{Z}_{\geq 0}$ and $a_i \in \mathcal{A}$ for all $-M \leq i \leq L$. The set of all Laurent polynomials in the indeterminate x with coefficients from the set $\mathcal{A}$ is denoted by $\mathcal{A}[x^{\pm 1}]$

Fact 1.1.1. *Let $\mathcal{F}$ be a field. The set of Laurent polynomials $\mathcal{F}[x^{\pm 1}]$ forms a ring.*

1.1.2 Matrix Notation

Throughout the thesis, matrices and vectors are represented by boldface letters. Commonly used matrix notations are summarized in Table 1.3. For each one of the matrices in this table, whenever the dimensions are clear from the context, we drop the subscripts explicitly indicating them.

Table 1.3: Matrix notations.

Symbol	*Meaning*
$\mathbf{A}^\dagger$	Hermitian transpose of the matrix $\mathbf{A}$
$\mathsf{det}(\mathbf{A})$	Determinant of the matrix $\mathbf{A}$
$\mathsf{wt}(a_1, \ldots, a_n)$	Weight of a vector $\mathbf{a} = [\, a_1, \ldots, a_n \,]^\dagger$ that is the number of nonzero components of $\mathbf{a}$
$\mathsf{diag}(a_1, \ldots, a_n)$	$n \times n$ diagonal matrix with $a_1, \ldots, a_n$ on the main diagonal
$_{m,n}$	$m \times n$ matrix with all entries being 1
$\mathbf{0}_{m,n}$	$m \times n$ matrix with all entries being 0
$\mathbf{I}_n$	$n \times n$ identity matrix (1's on the main diagonal and 0's elsewhere)
$\mathbf{e}_n^i$	i-th row of the $n \times n$ identity matrix $\mathbf{I}_n$

A matrix with polynomial entries is referred to as a *matrix polynomial* or a *polynomial matrix* [94]. These two terms, interchangeably used throughout the thesis, refer to the same mathematical object. Indeed, let $\mathbf{A}(x) \in \mathsf{M}_{\ell,m}(\mathcal{R}[x])$, where $\mathcal{R}$ is a ring, be a matrix with polynomial entries $a_{ij}(x) \in \mathcal{R}[x]$. The term "polynomial matrix" refers to the representation $[a_{ij}(x)]$ that emphasizes the matrix structure of $\mathbf{A}(x)$. On the other hand, the term "matrix polynomial" refers to the representation

$$\mathbf{A} = \mathbf{a}_0 + \mathbf{a}_1 x + \cdots + \mathbf{a}_L x^L \ , \qquad \mathbf{a}_L \neq \mathbf{0} \tag{1.2}$$

in which $\mathbf{a}_k \in \mathsf{M}_{\ell,m}(\mathcal{R})$, for all $k \in (\, L + 1 \,)$, is a matrix containing only the coefficients of the term x^k in all polynomials $a_{ij}(x)$. In other words, if $a_{ij}(x) = \sum_{k=0}^{L} a_{ijk} x^k$, where $a_{ijk} \in \mathcal{R}$, then $\mathbf{a}_k := [a_{ijk}]$. The integer $L \in \mathbb{Z}_{\geq 0}$ in (1.2) is the maximum order of all polynomial entries. Similar to ordinary polynomials, L is called the order of $\mathbf{A}(x)$ with respect to x and is denoted by $\mathsf{Ord}_x \mathbf{A}$. There is another nonnegative integer associated with the matrix polynomials in $\mathcal{A}[x^{-1}]$, i.e., polynomials with non-positive exponents. Every matrix polynomial in this polynomial set can be regarded as the transfer matrix of a system

[186]. The minimum number of delay elements with which the system can be constructed is called the *McMillan degree* or simply *degree* of the matrix polynomial. The degree of an arbitrary matrix polynomial $\mathbf{A}(x) \in \mathcal{A}[x^{-1}]$ with respect to the indeterminate x is denoted by $\mathsf{Deg}_x\mathbf{A}$. Although the degree and the order of a matrix polynomial $\mathbf{A}(x)$ could be different, the inequality

$$\mathsf{Deg}_x\mathbf{A} \leq \mathsf{Ord}_x\mathbf{A} \tag{1.3}$$

always holds. We note that the order and the degree of any polynomial are the same.

1.1.3 Asymptotic Notation

In this thesis, we only use the asymptotic lower bound O. Detailed descriptions can be found in [40]. Let $\mathfrak{D}$ be the set of all real-valued discrete functions from $\mathbb{N}$ to $\mathbb{R}_{\geq 0}$. For a given discrete function $f \in \mathfrak{D}$, we denote by $O(g(n))$ the *set of functions*

$$O(g(n)) := \Big\{ f \in \mathfrak{D} \,:\, \exists\, c \in \mathbb{R}_{>0},\, n_0 \in \mathbb{N} \text{ such that } 0 \leq f(n) \leq cg(n) \qquad \forall\, n \geq n_0 \Big\} \ . \tag{1.4}$$

A function f belongs to the set $O(g(n))$ if there exists a positive constant c such that $f(n) \leq cg(n)$ for sufficiently large n. Since $O(g(n))$ is a set, we could write $f \in O(g(n))$. Instead, with abuse of notation, we usually write "$f = O(g(n))$" to express the same notion.

1.1.4 General Notation

♢ The notations $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ are used for the ceiling and floor functions, respectively.

♢ The Kronecker delta is the function $\delta : \mathbb{Z} \to \{ 0, 1 \}$ defined as

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \ . \end{cases} \tag{1.5}$$

♢ For any $x \in \mathbb{Z}$ and any $N \in \mathbb{N}$, the notation $((x))_N$ denoted x mod N.

♢ Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a vector of variables and $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}^n$ be an integer vector. We use the short notation $\mathbf{x}^{\mathbf{a}} := x_1^{a_1} \cdots x_n^{a_n}$.

- ♢ Let $I = (i_0, \ldots, i_{d-1})$ be a d tuple. For any $j \in (\, d\,)$, the notation $I\langle j \rangle$ denotes a $d-1$ tuple obtained from I by removing i_j (the $(j+1)$-th coordinate). In case I is a set, it is regarded as a tuple with an arbitrary order.

- ♢ For an arbitrary $n \times l$ matrix $\mathbf{A}$ and any set of indices $I \subseteq [\, n\,]$, the symbol $\mathbf{A}\langle I \rangle$ represents a sub-matrix of $\mathbf{A}$ generated by removing any row of $\mathbf{A}$ with index outside I.

- ♢ The notation $x \| y$ implies the concatenation of mathematical objects x and y seen as an independent object of the same nature. Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ be the elements of x and y in the specified order, respectively. Then, the elements of the object $x \| y$ are $x_1, \ldots, x_n, y_1, \ldots, y_m$.

1.2 Linear Algebra Background

In this section, we review linear algebra concepts such as the involution, sesquilinear forms, unitary matrices, and paraunitary matrices. Throughout this section, we assume that $\mathcal{F}$ is an arbitrary field, $\mathbb{F}$ is a finite field, and $\mathcal{R}$ is an arbitrary ring.

1.2.1 Involution

From the space of all linear functions from $\mathcal{R} \to \mathcal{R}$, we fix an *involution* "$-$" that, by its definition, has the property $\overline{\overline{a}} = a$ for all $a \in \mathcal{R}$. For example, in the complex field, the conjugation operation is an involution. In finite fields, the identity operator is trivially an involution. In the following, we progressively extend the definition of involution to higher algebraic structures.

We naturally extend the concept of involution to the ring $\mathcal{R}[x]$ of univariate polynomials over $\mathcal{R}$ as follows.

Definition 1.2.1 (Involution "$-$" over the Ring of Univariate Polynomials). *We define the*

involution "$-$" over the polynomial ring $\mathcal{R}[x]$ as follows

$$\begin{array}{rcccc} - & : & \mathcal{R}[x] & \longrightarrow & \mathcal{R}[x^{-1}] \\ & & a(x) = \sum_{i=0}^{L} a_i x^i & \longmapsto & \overline{a}(x) = \sum_{i=0}^{L} \overline{a}_i x^{-i} \ . \end{array}$$

REMARK. *By this definition, the involution "$-$" over the polynomial ring $\mathbb{F}[x]$ is simplified to $a(x) = \sum_i a_i x^i \mapsto \overline{a}(x) = \sum_i a_i x^{-i}$ for all $a \in \mathbb{F}[x]$.*

REMARK. *Over the module $\mathcal{R}^*$ of finite-length sequences over the ring $\mathcal{R}$, we define "$-$" in a similar fashion. For every $a = (\, a_n \,) \in \mathcal{R}^*$, we define $\overline{a} := (\, \overline{a}_n \,) \in \mathcal{R}^*$.*

Given an involution and an arbitrary polynomial, we can uniquely associate to it a reciprocal polynomial defined as follows [198].

DEFINITION 1.2.2 (Reciprocal Polynomial). *Given an involution "$-$", we define the reciprocity operation as*

$$\begin{array}{rcccc} \mathsf{R} & : & \mathcal{R}[x] & \longrightarrow & \mathcal{R}[x] \\ & & a(x) = \sum_{i=0}^{L} a_i x^i & \longmapsto & a^{\mathsf{R}}(x) = \sum_{i=0}^{L} \overline{a}_{L-i} x^i. \end{array}$$

REMARK. *It can be easily seen that $a^{\mathsf{R}}(x) = x^L\, \overline{a}(\overline{x}^{-1})$ for any degree-L polynomial $a \in \mathcal{R}[x]$.*

REMARK. *Over the polynomial ring $\mathbb{F}[x]$, the reciprocity operation is simplified to $a(x) = \sum_{i=0}^{L} a_i x^i \mapsto a^{\mathsf{R}}(x) = \sum_{i=0}^{L} a_{L-i} x^i$ for all $a \in \mathbb{F}[x]$. Therefore, $a^{\mathsf{R}}(x) = x^L\, a(x^{-1})$.*

As the next step, we extend our definition of "$-$" to the module $\mathsf{M}_{M,N}(\mathcal{R}[x])$ consisting of matrices with polynomial entries. To be consistent with the notation used in mathematical textbooks, we employ the symbol † in the extended definition.

DEFINITION 1.2.3 (Involution † over the Module of Univariate Matrix Polynomials). *We define the involution † over the matrix module $\mathsf{M}_{M,N}(\mathcal{R}[x])$ as*

$$\begin{array}{rcccc} \dagger & : & \mathsf{M}_{M,N}(\mathcal{R}[x]) & \longrightarrow & \mathsf{M}_{N,M}(\mathcal{R}[x^{-1}]) \\ & & \mathbf{A} = [a_{ij}] & \longmapsto & \mathbf{A}^{\dagger} := [a'_{ij}] \ , \end{array}$$

where $a'_{ij} := \overline{a}_{ji}$ for all $i \in [\, M \,]$ and all $j \in [\, N \,]$.

REMARK. *In this book, † over the matrix module* $\mathsf{M}_{M,N}(\mathbb{F}[x])$ *is extensively used. For every* $\mathbf{A}(x) \in \mathsf{M}_{M,N}(\mathbb{F}[x])$*, we have* $\mathbf{A}^{\dagger}(x) = \mathbf{A}^{T}(x^{-1}) \in \mathsf{M}_{N,M}(\mathbb{F}[x^{-1}])$*, where the superscript* T *denotes matrix transposition.*

REMARK. *Over the module of constant matrices, we have* $\mathbf{A}^{\dagger} = [\,\overline{a}_{ji}\,] \in \mathsf{M}_{N,M}(\mathcal{R})$ *for every* $\mathbf{A} = [\,a_{ij}\,] \in \mathsf{M}_{M,N}(\mathcal{R})$*.*

1.2.2 Sesquilinear Form

Let $M \in \mathbb{N}$ be an arbitrary but fixed integer and $- : \mathcal{R} \to \mathcal{R}$ be an involution. As defined in [122, 199], we say that a map $\langle \cdot, \cdot \rangle : \mathcal{R}^M \times \mathcal{R}^M \to \mathcal{R}$ is a *sesquilinear form*[1] if for arbitrary $\alpha \in \mathcal{R}$, $\mathbf{a} = [\,a_i\,] \in \mathcal{R}^M$, and $\mathbf{b} = [\,b_i\,] \in \mathcal{R}^M$, we have

$$\langle \alpha\mathbf{a},\, \mathbf{b} \rangle = \alpha\, \langle \mathbf{a},\, \mathbf{b} \rangle \quad , \qquad\qquad \langle \mathbf{a},\, \alpha\mathbf{b} \rangle = \overline{\alpha}\, \langle \mathbf{a},\, \mathbf{b} \rangle \quad .$$

For example, consider the module $(\mathbb{F}[x])^M$. It can be easily verified that the following map is a sesquilinear form.

$$\langle \mathbf{u}(x),\, \mathbf{v}(x) \rangle := \mathbf{u}^{\dagger}(x)\, \mathbf{v}(x) \qquad \forall \mathbf{u}(x), \mathbf{v}(x) \in (\mathbb{F}[x])^2 \tag{1.6}$$

Given any sesquilinear form $\langle \cdot, \cdot \rangle$, we can define the *conjugate transpose* sesquilinear form $\langle \cdot, \cdot \rangle_c$ defined by $\langle \mathbf{a},\, \mathbf{b} \rangle_c := \overline{\langle \mathbf{b},\, \mathbf{a} \rangle}$. A symmetric sesquilinear form, a form being identical to is conjugate, is called a *Hermitian form*. A sesquilinear form in which the corresponding involution is the identity map is called a *bilinear form*. As its name implies, a bilinear form is linear with respect to both input arguments. We note that a bilinear form is not necessarily symmetric.

To show an example of a symmetric bilinear form, consider the map

$$\langle \mathbf{a},\, \mathbf{b} \rangle := \mathbf{a}^{\dagger}\, \mathbf{b} \qquad \forall \mathbf{a}, \mathbf{b} \in \mathbb{F}^M \quad . \tag{1.7}$$

It can be easily verified that this is a symmetric bilinear form. Similar to (1.7), we can define a symmetric bilinear form over the vector space $\mathbb{F}^*$ of all finite sequences over any

[1] Sesquilinear means $1\frac{1}{2}$ times linear.

finite field $\mathbb{F}$. Precisely, we define

$$\langle a,\, b\rangle := \sum_{n\in\mathcal{I}} a_n\, b_n \qquad a, b \in \mathbb{F}^* \ , \tag{1.8}$$

where $\mathcal{I}$ is a countable set used to index sequences in $\mathbb{F}^*$. We emphasize that since all sequences in $\mathbb{F}^*$ are finite, the summation in (1.8) is actually a finite sum. Therefore, the symmetric bilinear form in (1.8) is well defined.

A sesquilinear form over a module $\mathcal{M}$ that satisfies the property

$$\langle \mathbf{a},\, \mathbf{a}\rangle = 0 \qquad \Leftrightarrow \qquad \mathbf{a} = \mathbf{0} \qquad \forall \mathbf{a} \in \mathcal{M} \ , \tag{1.9}$$

called *positive definiteness*, is called an *inner product* or a *dot product*. The notion of orthogonality is technically only associated with inner products. However, we use it for sesquilinear forms as well. Two vectors $\mathbf{a}, \mathbf{b} \in \mathcal{M}$ are called *orthogonal*, denoted by $\mathbf{a} \perp \mathbf{b}$, if and only if $\langle \mathbf{a},\, \mathbf{b}\rangle = \mathbf{0}$. A set of vectors $\{\, \mathbf{a}_1, \ldots, \mathbf{a}_\lambda \,\} \subset \mathcal{M}$ is called *orthogonal* if $\langle \mathbf{a}_i,\, \mathbf{a}_j\rangle = 0$ for all distinct $i, j \in [\,\lambda\,]$.

Let $\mathcal{N} \subset \mathcal{M}$ be a submodule. A vector $\mathbf{v} \in \mathcal{M}$ is a *right null vector* of $\mathcal{N}$ with respect to the sesquilinear form $\langle \cdot, \cdot \rangle$ if it is orthogonal to all vectors in $\mathcal{N}$ from the right, i.e., $\langle \mathbf{w},\, \mathbf{v}\rangle = 0$ for all $\mathbf{w} \in \mathcal{N}$. In a similar way, we define a *left null vector*. The set of all right null vectors of $\mathcal{N}$ is called the *right null space* of $\mathcal{N}$ and is denoted as $\mathsf{Null}_R(\mathcal{N})$, i.e.,

$$\mathsf{Null}_R(\mathcal{N}) := \Big\{ \mathbf{v} \in \mathcal{M} :\ \langle \mathbf{w},\, \mathbf{v}\rangle = 0 \qquad \forall \mathbf{w} \in \mathcal{N} \Big\} \ . \tag{1.10}$$

In a similar fashion, we define the left null space of $\mathcal{N}$ that we denote by $\mathsf{Null}_L(\mathcal{N})$. One easily verifies that the left and right null spaces are, in fact, modules. A null space is *self-dual* if it is equal to the vector space itself. For example if $\mathsf{Null}_R(\mathcal{N}) = \mathcal{N}$, then the right null space of $\mathcal{N}$ is self-dual.

In some cases, the left and right null spaces are the same. This situation occurs, for example, when the underlying sesquilinear form is symmetric. The sesquilinear form defined in (1.7) satisfies this property. By the *null space of a matrix*, we mean the null space of the module generated by the linear span of its columns. To put this definition in mathematical

language, let $\mathbf{A} \in \mathsf{M}_M(\mathcal{R})$ be a matrix and $\mathbf{a}_1, \ldots, \mathbf{a}_N \in \mathcal{R}^M$ be its column vectors. Then,

$$\mathsf{Null}(\mathbf{A}) := \mathsf{Null}\Big(\mathsf{span}\,\{\,\mathbf{a}_1, \ldots, \mathbf{a}_M\,\}\Big)\ . \tag{1.11}$$

FACT 1.2.1. *Let* $\mathbf{A} \in \mathsf{M}_M(\mathcal{R})$ *be an arbitrary matrix and* $\mathbf{v} \in \mathcal{R}^M$ *an arbitrary vector. Then,* $\mathbf{v} \in \mathsf{Null}(\mathbf{A})$ *if and only if* $\mathbf{A}^\dagger \mathbf{v} = \mathbf{0}$.

Another notion, which we borrow from inner products, is norm. The map

$$\begin{array}{rccc} \|\cdot\| & : & \mathcal{M} & \longrightarrow & \mathcal{R} \\ & & \mathbf{a} & \longmapsto & \langle \mathbf{a}, \mathbf{a} \rangle \end{array} \tag{1.12}$$

is called the *norm*. A vector $\mathbf{a} \in \mathcal{M}$ with $\|\mathbf{a}\| = 0$ is called *self-orthogonal*. A set of vectors $\{\,\mathbf{a}_1, \ldots, \mathbf{a}_\lambda\,\} \subset \mathcal{M}$ is called *orthonormal* if it is an orthogonal set with the additional property $\|\mathbf{a}_i\| = 1$ for all $i \in [\,\lambda\,]$.

FACT 1.2.2. *Let* $\{\,\mathbf{a}_1, \ldots, \mathbf{a}_\lambda\,\} \subset \mathcal{M}$ *be an orthogonal set. Then, for any linear combination* $\mathbf{x} = \alpha_1 \mathbf{a}_1 + \cdots + \alpha_\lambda \mathbf{a}_\lambda \in \mathcal{M}$ *with* $\alpha_1, \ldots, \alpha_\lambda \in \mathcal{R}$, *we have*

$$\|\mathbf{x}\| = \sum_{i=1}^{\lambda} \alpha_i \overline{\alpha}_i \|\mathbf{a}_i\|\ .$$

1.2.3 Unitary Matrix

Unitary matrices are a subclass of the general linear group $\mathsf{GL}(\mathcal{R})$. Let $\dagger$ be the involution in Definition 1.2.3 and $\langle \cdot, \cdot \rangle$ be the sesquilinear form in (1.7). We define unitary matrices as follows.

DEFINITION 1.2.4 (Unitary Matrix). *A matrix* $\mathbf{A} \in \mathsf{M}_M(\mathcal{R})$ *is unitary if and only if* $\mathbf{A}^\dagger \mathbf{A} = \mathbf{I}$.

The set of all $M \times M$ unitary matrices over the ring $\mathcal{R}$ is denoted by $\mathsf{U}_M(\mathcal{R})$. One easily verifies that $\mathsf{U}_M(\mathcal{R})$ with the ordinary matrix multiplication forms a group. As an example of unitary matrices, consider the matrix $\mathbf{A} = \mathbf{I} + \zeta \mathbf{v}\mathbf{v}^\dagger \in \mathsf{M}_M(\mathbb{F})$ in which $\zeta \in \mathbb{F}$ is an arbitrary constant and $\mathbf{v} \in \mathbb{F}^M$ is a vector satisfying the property $\|\mathbf{v}\| = 0$. This matrix

is unitary since

$$\begin{aligned}\mathbf{A}^\dagger\mathbf{A} &= \left(\mathbf{I}+\zeta\mathbf{v}\mathbf{v}^\dagger\right)\left(\mathbf{I}+\zeta\mathbf{v}\mathbf{v}^\dagger\right)\\ &= \mathbf{I}+\zeta\mathbf{v}\mathbf{v}^\dagger+\zeta\mathbf{v}\mathbf{v}^\dagger+\zeta^2\mathbf{v}\left(\mathbf{v}^\dagger\mathbf{v}\right)\mathbf{v}^\dagger\\ &= \mathbf{I}\ . \qquad \text{(Recall that } \mathbb{F} \text{ is a field with characteristic two.)}\end{aligned}$$

To provide an example of unitary matrices over the complex field, consider the matrix

$$\mathbf{H}_\mathbf{u} := \mathbf{I} - 2\mathbf{u}\mathbf{u}^\dagger\ , \tag{1.13}$$

where $\mathbf{u} \in \mathbb{C}^M$ such that $\|\mathbf{u}\| = 1$. It can be easily verified that this is a unitary matrix [186]. The matrix in (1.13) is called a *Householder matrix*.

By Definition 1.2.4, if $\mathbf{A}$ is unitary, then $\mathbf{A}^{-1} = \mathbf{A}^\dagger$. This result leads to $\mathbf{A}\mathbf{A}^\dagger = \mathbf{I}$. In fact, one can easily prove the following fact [161].

FACT 1.2.3. *For any matrix* $\mathbf{A} = [\,\mathbf{a}_1, \ldots, \mathbf{a}_M\,] \in \mathsf{M}_M(\mathcal{R})$*, where* $\mathbf{a}_1, \ldots, \mathbf{a}_M$ *are its columns, the following statements are equivalent.*

(i) $\mathbf{A} \in \mathsf{U}_M(\mathcal{R})$

(ii) $\mathbf{A}^\dagger \in \mathsf{U}_M(\mathcal{R})$

(iii) $\langle \mathbf{a}_i, \mathbf{a}_j \rangle = \delta(i-j) \qquad \forall i, j \in [\,M\,]$

(iv) $\mathbf{A}^\dagger\mathbf{A} = \mathbf{I}$

(v) $\mathbf{A}\mathbf{A}^\dagger = \mathbf{I}$

(vi) $\mathbf{A}^{-1} = \mathbf{A}^\dagger$

An interesting property of unitary matrices is that they preserve the norm of any vector. This important result is stated in the following.

LEMMA 1.2.1 (Norm Preservation of Unitary Matrices). *Let* $\mathbf{A} \in \mathsf{U}_M(\mathcal{R})$ *be an arbitrary unitary matrix and* $\mathbf{v} \in \mathcal{R}^M$ *an arbitrary vector for some* $M \in \mathbb{N}$*. The unitary matrix preserves the norm of* $\mathbf{v}$*, with respect to the symmetric bilinear form (1.7), as a linear transformation, i.e.,* $\|\mathbf{A}\mathbf{v}\| = \|\mathbf{v}\|$*.*

Proof. $\|\mathbf{A}\mathbf{v}\| = \mathbf{v}^\dagger\mathbf{A}^\dagger\mathbf{A}\mathbf{v} = \mathbf{v}^\dagger\mathbf{v} = \|\mathbf{v}\|$. ∎

1.2.4 Paraunitary Matrix

A natural generalization of unitary matrices is paraunitary (PU) matrices. These are a subclass of the general linear group of all invertible polynomial matrices $\mathsf{GL}(\mathcal{R}[x])$.

DEFINITION 1.2.5 (PU Matrix). *A polynomial matrix $\mathbf{P} \in \mathsf{M}_M(\mathcal{R}[x])$ is PU if and only if $\mathbf{P}^\dagger\mathbf{P} \equiv \mathbf{I}$. This is equivalent to*

$$\mathbf{P}^\dagger(x)\mathbf{P}(x) = \mathbf{I} \qquad \forall x \in \mathcal{R} \setminus \{\,0\,\} \quad .$$

The set of all $M \times M$ PU matrices over the polynomial module $\mathcal{R}[x]$ is denoted by $\mathsf{PU}_M(\mathcal{R}[x])$. One easily verifies that $\mathsf{PU}_M(\mathcal{R}[x])$ with the ordinary matrix multiplication forms a group. As an example, consider the matrix polynomial $\mathbf{P}(x) = \mathbf{I} + \mathbf{v}\mathbf{v}^\dagger + \mathbf{v}\mathbf{v}^\dagger x \in \mathsf{M}_M(\mathbb{F}[x])$ in which $\mathbf{v} \in \mathbb{F}^M$ such that $\|\mathbf{v}\| = 1$. It can be easily verified that this is a PU matrix. Similar to Fact 1.2.3, we can prove the following.

FACT 1.2.4. *For any polynomial matrix $\mathbf{P}(x) = [\,\mathbf{p}_1(x), \ldots, \mathbf{p}_M(x)\,] \in \mathsf{M}_M(\mathcal{R}[x])$, where $\mathbf{p}_1(x), \ldots, \mathbf{p}_M(x)$ are its columns, the following statements are equivalent.*

(i) $\mathbf{P}(x) \in \mathsf{PU}_M(\mathcal{R}[x])$

(ii) $\mathbf{P}(x) \in \mathsf{PU}_M(\mathcal{R}[x^{-1}])$

(iii) $\langle \mathbf{p}_i,\, \mathbf{p}_j \rangle = \delta(i-j) \qquad \forall i, j \in [\,M\,]$

(iv) $\mathbf{P}^\dagger(x)\mathbf{P}(x) \equiv \mathbf{I}$

(v) $\mathbf{P}(x)\mathbf{P}^\dagger(x) \equiv \mathbf{I}$

(vi) $\mathbf{P}^{-1}(x) = \mathbf{P}^\dagger(x)$

Similar to unitary matrices, PU matrices have the norm preservation property as well.

LEMMA 1.2.2 (Norm Preservation of PU Matrices). *Let $\mathbf{P}(x) \in \mathsf{PU}_M(\mathcal{R}[x])$ be an arbitrary PU matrix and $\mathbf{v}(x) \in (\mathcal{R}[x])^M$ an arbitrary vector polynomial for some $M \in \mathbb{N}$. The PU matrix preserves the norm of $\mathbf{v}(x)$, with respect to the symmetric bilinear form (1.7), as a linear transformation, i.e., $\|\mathbf{P}(x)\mathbf{v}(x)\| = \|\mathbf{v}(x)\|$.*

Proof. $\|\mathbf{P}(x)\mathbf{v}(x)\| = \mathbf{v}^\dagger(x)\mathbf{P}^\dagger(x)\mathbf{P}(x)\mathbf{v}(x) = \mathbf{v}^\dagger(x)\mathbf{v}(x) = \|\mathbf{v}(x)\|$. ∎

Part I

Bivariate PU Filter Banks over Fields of Characteristic Two

CHAPTER 2

Finite-Field Wavelet Transform

Wavelets and filter banks over real or complex fields have found many applications such as digital audio and image processing [189, 183, 184]. The extension of univariate filter banks onto finite fields have been studied by several authors [185, 161, 86, 30, 84]. They have applications in both error-control coding [86, 171, 172] and cryptography [60, 63, 33, 55]. Motivated by these applications, the primary objective of Part I is to propose a method for efficiently generating bivariate PU matrix polynomials over fields of characteristic two. Prior to providing such method, a review of finite-field wavelets and also univariate PU matrices is required. In Section 2.1, we briefly review the mathematical philosophy behind the wavelet transform and explain its connection with filter banks. In addition, we explain the importance of PU matrices in the design of finite impulse response (FIR) filter banks. In the direction of reviewing different building blocks proposed for generating univariate PU matrix polynomials, we review the factorization of unitary matrices in Section 2.2. Followed by that, we introduce univariate PU building block in Section 2.3.

2.1 Quick Introduction to Wavelet Transform

Wavelets have a very rich history in mathematics, engineering, and applied sciences [49, 51, 180, 186, 190, 83]. They are established as powerful tools for many practical problems. In contrast to the Fourier transform that mostly suits processing period discrete- or continuous-time signals, the wavelet transform can be used to study signals with finite supports. Another distinction between these two transforms is that the Fourier transform does not exist over all finite fields since it requires an element with a specific order. In the

other hand, the discrete-time wavelet transform (DTWT) expands a discrete-time sequence as a linear combination of basis sequences belonging to nesting orthogonal-complement vector spaces [85].

To mathematically explain the DTWT, let $\mathcal{V}$ be a (finite or infinite dimensional) vector space over a finite field $\mathbb{F}$. The wavelet transform decomposes $\mathcal{V}$ into two orthogonally complement vector spaces $\mathcal{V}_1$ and $\mathcal{W}_1$ such that $\mathcal{V} = \mathcal{V}_1 \oplus \mathcal{W}_1$, where the binary operator $\oplus$ is the direct sum. Let $\{\, \varphi_\ell \in \mathbb{F}^* \,:\, \forall \ell \in \mathbb{Z} \,\}$ and $\{\, \psi_\ell \in \mathbb{F}^* \,:\, \forall \ell \in \mathbb{Z} \,\}$ be the basis sets of the vector subspaces $\mathcal{V}_1$ and $\mathcal{W}_1$, respectively, in which $\varphi_\ell(n) = \varphi(n-2\ell)$ and $\psi_\ell(n) = \psi(n-2\ell)$ for all $\ell, n \in \mathbb{Z}$. Here, $\varphi \in \mathbb{F}^*$ and $\psi \in \mathbb{F}^*$ are respectively called the *scaling sequence* and the *mother wavelet*. In orthogonal wavelet transforms, they must satisfy the following properties.

$$\langle \varphi_k,\, \varphi_\ell \rangle = \delta(k-\ell) \qquad \forall k, \ell \in \mathbb{Z} \tag{2.1a}$$

$$\langle \psi_k,\, \psi_\ell \rangle = \delta(k-\ell) \qquad \forall k, \ell \in \mathbb{Z} \tag{2.1b}$$

$$\langle \varphi_k,\, \psi_\ell \rangle = 0 \qquad \forall k, \ell \in \mathbb{Z} \tag{2.1c}$$

An important consequence of these relations is the following lemma proved in [83].

LEMMA 2.1.1. *Over any (finite or infinite dimensional) field $\mathbb{F}$, the scaling function $\varphi \in \mathbb{F}^*$ and the mother wavelet $\psi \in \mathbb{F}^*$ have both supports with even lengths.*

Any sequence $x \in \mathcal{V}$ can be expanded into a linear combination of basis sequences as[1]

$$x(n) = \sum_{\ell=-\infty}^{\infty} w_0(\ell)\, \varphi_\ell(n) + \sum_{\ell=-\infty}^{\infty} w_1(\ell)\, \psi_\ell(n) \;\;, \tag{2.2}$$

where $w_0, w_1 \in \mathbb{F}^*$ are the *wavelet coefficients* that are uniquely determined as

$$w_0(\ell) = \langle x,\, \varphi_\ell \rangle \qquad \forall \ell \in \mathbb{Z} \tag{2.3a}$$

$$w_1(\ell) = \langle x,\, \psi_\ell \rangle \qquad \forall \ell \in \mathbb{Z} \;\;. \tag{2.3b}$$

Equations (2.3) and (2.2) define the DTWT and its inverse, respectively.

[1]We note that the notion of convergence over finite fields is meaningless. Every summation in this paper is taken over a finite set of indices since all sequences in $\mathbb{F}^*$ have finite supports.

The DTWT and its inverse are already recognized as equivalent to the analysis and synthesis banks of a two-band filter bank as shown in Figure 2.1. It can be easily verified that for the analysis bank in this structure to output the sequences w_0 and w_1, we must have $g_0(n) = \varphi(-n)$ and $g_1(n) = \psi(-n)$ for all $n \in \mathbb{Z}$. To avoid anti-causal filters, we design the scaling sequence and the mother wavelet such that their support is the set $\{ -L, -L+1, \ldots, 0 \}$ for some $L \in \mathbb{Z}_{\geq 0}$. The inverse DTWT can also be realized using the synthesis bank of the filter bank in Figure 2.1. The output of this system is a shifted version of x in which the nonnegative delay is introduced by the filters h_0 and h_1. The equation representing the sequence reconstruction in terms of the analysis filters h_0 and h_1 is

$$x(n-L) = \sum_{\ell=-\infty}^{\infty} w_0(\ell)\, h_0(n-2\ell-L) + \sum_{\ell=-\infty}^{\infty} w_1(\ell)\, h_1(n-2\ell-L) \ , \tag{2.4}$$

where $h_0(n) = \varphi(n-L)$ and $h_1(n) = \psi(n-L)$, for all $n \in \mathbb{Z}$, are causal filters.

A two-band filter bank that realizes a DTWT and its inverse (i.e., it perfectly reconstructs the input sequence x from the wavelet coefficients w_0 and w_1) is said to possess the perfect reconstruction (PR) property. For the filter bank in Figure 2.1 to achieve the PR property, the filters g_i and h_i must satisfy some conditions. To obtain them, we use the equivalent polyphase representation of the filter bank in Figure 2.2. In this figure, $\mathbf{P}(z) = [\, P_{ij}(z) \,] \in \mathsf{M}_2(\mathbb{F}[z^{-1}])$ and $\mathbf{Q}(z) = [\, Q_{ij}(z) \,] \in \mathsf{M}_2(\mathbb{F}[z^{-1}])$ are the polyphase matrices if the analysis and synthesis banks, respectively. To express their relationships with the filters in Figure 2.1, let $G_i(z) := \sum_{n=0}^{L} g_i(n)\, z^{-n} \in \mathbb{F}[z^{-1}]$ and $H_i(z) := \sum_{n=0}^{L} h_i(n)\, z^{-n} \in \mathbb{F}[z^{-1}]$ be

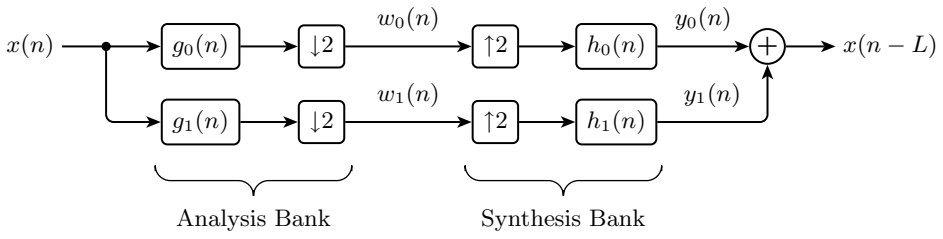


Figure 2.1: Two-band filter bank.

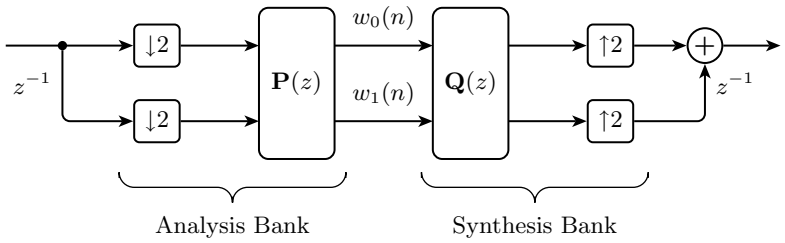


Figure 2.2: Polyphase representation of two-band filter bank.

the z-transforms of g_i and h_i, respectively, for both $i = 0, 1$. Using this notation, we have

$$G_i(z) = P_{i0}(z^2) + z^{-1}\, P_{i1}(z^2) \tag{2.5a}$$

$$H_i(z) = z^{-1}\, Q_{0i}(z^2) + Q_{1i}(z^2) \ . \tag{2.5b}$$

It is shown in [186] that the possession of the PR property can be expressed in the compact form

$$\mathbf{Q}(z)\,\mathbf{P}(z) = z^{\theta}\,\mathbf{I} \ , \tag{2.6}$$

where $\theta \in \mathbb{Z}$. To avoid infinite impulse response (IIR) filters, the polyphase matrix $\mathbf{P}(z)$ is chosen to be PU. With this choice, it suffices to set $\mathbf{Q}(z) = z^{\theta}\,\mathbf{P}^{\dagger}(z)$ to achieve the PR property.

The following fact, proved in [85] about the filter banks, will be used later in Chapter 5.

FACT 2.1.1. *When the polyphase matrices of a two-channel filter bank as in Figure 2.1 are all PU, the following hold.*

1. *Polynomials $G_i(z)$ and $H_i(z)$ all have the same degree $L \in \mathbb{N}_{\mathsf{odd}}$, which is an odd integer. Moreover, their constant terms are nonzero, i.e., $g_i(0) \neq 0$ and $h_i(0) \neq 0$.*
2. *$G_1(z) = z^{-L}\, G_0(z)$ and $g_1(n) = g_0(L - n)$ for all $n \in \{\, 0, 1, \ldots, L \,\}$.*

Collectively, to realize a DTWT and its inverse using the two-band filter bank in Figure 2.1, one employs Algorithm 2.1. By this algorithm, the main issue in the construction of the DTWT is the generation of PU matrices. To this end, in the rest of this chapter, we review the univariate PU building blocks for the generation of univariate PU matrices.

Algorithm 2.1: DTWT construction.

INPUT: Matrix polynomial $\mathbf{P}(z) \in \mathsf{PU}_2(\mathbb{F}[z^{-1}])$

OUTPUT: Causal filters g_0, g_1, h_0, and h_1

1. $\theta \leftarrow \mathsf{Ord}_{z^{-1}}\mathbf{P}$
2. $\mathbf{Q}(z) \leftarrow z^{-\theta}\,\mathbf{P}^{\dagger}(z)$
3. Obtain $G_i(z)$ using (2.5a)
4. Obtain $H_i(z)$ using (2.5b)

2.2 Unitary Matrices

By Definition 1.2.4, a matrix $\mathbf{A} \in \mathsf{M}_M(\mathbb{F})$, for any $M \in \mathbb{N}$, is unitary if and only if $\mathbf{A}^{\dagger}\,\mathbf{A} = \mathbf{I}$. These matrices are important in the construction of the PU matrices. Furthermore, they will be used in the construction of the public-key cryptosystem PAC in Chapter 6 in Part II.

The building block that generates all unitary matrices over $\mathbb{F}$ is the matrix

$$\mathbf{U}_{\zeta,\mathbf{u}} := \mathbf{I} + \zeta\mathbf{u}\mathbf{u}^{\dagger} \tag{2.7}$$

in which $\|\mathbf{u}\| = 0$ and $\zeta \in \mathbb{F}$ is an arbitrary element. The fact that this matrix generates all unitary matrices is summarized in the following theorem for future references.

THEOREM 2.2.1 (Factorization of Unitary Matrices). *Let $\mathbf{A} \in \mathsf{M}_M(\mathbb{F})$ be an arbitrary matrix. Then, $\mathbf{A}$ is unitary if and only if it can be factored as*

$$\mathbf{A} = \mathbf{U}_{\zeta_M,\mathbf{u}_M}\,\mathbf{U}_{\zeta_{M-1},\mathbf{u}_{M-1}}\cdots\mathbf{U}_{\zeta_2,\mathbf{u}_2}\,\mathbf{S} \tag{2.8}$$

in which $\mathbf{S}$ is a permutation of the identity matrix, $\zeta_i \in \mathbb{F}$, and $\mathbf{u}_i \in \mathbb{F}^M$ with the property $\|\mathbf{u}_i\| = 0$ for all $i = 2, \ldots, M$.

The proof is provided in [85, 83].

2.3 Univariate PU Matrices

A matrix polynomial $\mathbf{P}(z) \in \mathsf{M}_M(\mathbb{F}[z^{-1}])$ is PU if and only if $\mathbf{P}^{\dagger}(z)\,\mathbf{P}(z) = \mathbf{I}$ (Definition 1.2.5). Based on the size of the univariate PU matrix (i.e., the value of M), there are

different generating building blocks. This is in contrast to the complex field over which univariate PU matrices are generated with a single building block [186]. In the rest of this section, we list all PU building blocks, and then, in a theorem, explain the ones that generate 2×2 PU matrices and those that generate $M \times M$ ones for $M \geq 3$.

The univariate PU building blocks are as follows.

$\diamondsuit$ Degree-one building block

$$\mathbf{B}_1(z;\, \mathbf{v}) := \mathbf{I} + \mathbf{v}\mathbf{v}^\dagger + \mathbf{v}\mathbf{v}^\dagger z^{-1} \quad \in \mathsf{PU}_M(\mathbb{F}[z^{-1}]) \ , \tag{2.9}$$

where $\mathbf{v} \in \mathbb{F}^M$ is a unit-norm vector, i.e., $\|\mathbf{v}\| = 1$.

$\diamondsuit$ Degree-two building block

$$\mathbf{B}_2(z;\, \mathbf{u}, \mathbf{v}) := \mathbf{I} + \mathbf{u}\mathbf{v}^\dagger + \mathbf{v}\mathbf{u}^\dagger + (\mathbf{u}\mathbf{v}^\dagger + \mathbf{v}\mathbf{u}^\dagger)z^{-1} \quad \in \mathsf{PU}_M(\mathbb{F}[z^{-1}]) \tag{2.10}$$

in which $\mathbf{u}, \mathbf{v} \in \mathbb{F}^M$ are self-orthogonal vectors, i.e., $\|\mathbf{u}\| = \|\mathbf{v}\| = 0$, such that $\mathbf{u}^\dagger\, \mathbf{v} = 1$.

$\diamondsuit$ Degree-2τ building block

$$\mathbf{S}_{2\tau}(z;\, \zeta) := \zeta \ \ {}_{2,2} + \mathbf{I} z^{-\tau} + \zeta \ \ {}_{2,2} z^{-2\tau} \quad \in \mathsf{PU}_2(\mathbb{F}[z^{-1}]) \tag{2.11}$$

in which $\tau \in \mathbb{N}$ and $\zeta \in \mathbb{F}$ is an arbitrary finite-field element.

$\diamondsuit$ Degree-$M\tau$ building block

$$\mathbf{R}_{M\tau}(x;\, \mathbf{V}, \mathbf{\Lambda}) := \mathbf{V}\, \mathbf{\Lambda}\, \mathbf{V}^\dagger + \mathbf{I} x^{-\tau} + \mathbf{V}\, \mathbf{\Lambda}\, \mathbf{V}^\dagger x^{-2\tau} \quad \in \mathsf{PU}_M(\mathbb{F}[z^{-1}]) \ . \tag{2.12}$$

Here, $\tau \in \mathbb{N}$, $\mathbf{\Lambda} = \mathsf{diag}(\lambda_1, \ldots, \lambda_M) \in \mathsf{M}_M(\mathbb{F})$, and $\mathbf{V} \in \mathsf{M}_M(\mathbb{F})$. If $\mathbf{v}_1, \ldots, \mathbf{v}_M$ are the column vectors of $\mathbf{V}$, then $\mathbf{v}_i^\dagger\, \mathbf{v}_j = 0$ for all $i, j \in [\, M\,]$.

It is worth noting that the product term $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\dagger$ in definition (2.12) is independent of the order by which we arrange the columns of $\mathbf{V}$ as long as they are consistent with the order of elements on the main diagonal of $\mathbf{\Lambda}$. In fact, we have

$$\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\dagger = \sum_{i=1}^{M} \lambda_i \mathbf{v}_i \mathbf{v}_i^\dagger \ . \tag{2.13}$$

This results suggests choosing distinct vectors for the columns of $\mathbf{V}$.

The following theorem, proved in [85, 83], states that these building blocks generate all univariate PU matrices.

THEOREM 2.3.1 (Univariate PU Factorization). *Let $\mathbf{P}(z) \in \mathsf{M}_M(\mathbb{F}[z^{-1}])$ be an arbitrary polynomial matrix. This matrix is PU if and only if it can be factorized as*

$$\mathbf{P}(z) = z^m \prod_{i=1}^{N} \mathbf{A}_i \, \mathbf{C}_i(z)$$

in which $m \in \mathbb{Z}$ and $N \in \mathbb{N}$ such that $N \leq \mathsf{Deg}_z \mathbf{P}$. In addition, for all $i \in [\, N \,]$, $\mathbf{A}_i$ is either the identity matrix or a unitary matrix and $\mathbf{C}_i(z) \in \mathsf{PU}_M(\mathbb{F}[z^{-1}])$ is one of the introduce univariate building blocks depending on the size of $\mathbf{P}(z)$.

- *$M = 2$: In this case, $\mathbf{C}_i(z)$ is either the degree-one building block $\mathbf{B}_1$ in (2.9) or the degree-2τ building block $\mathbf{S}_{2\tau}$ in (2.11).*
- *$M \geq 3$: $\mathbf{C}_i(z)$ is either the degree-one building block $\mathbf{B}_1$ in (2.9), the degree-two building block $\mathbf{B}_2$ in (2.10), or the degree-$M\tau$ building block $\mathbf{R}_{M\tau}$ in (2.12).*

Based on this theorem, one can generate univariate PU matrices by arbitrarily multiplying appropriate building blocks. In the applications of interest in this thesis, including unitary matrices or pure delays in the construction of PU matrices only adds to the computational complexity without serving any practical purpose. Hence, in the following, we provide algorithms for generating univariate PU matrices in which only the introduced univariate building blocks are used.

By Theorem 2.3.1, to construct a univariate 2×2 PU matrix, the following parameters must be known to the designer.

– $\beta_1, \beta_2 \in \mathbb{Z}_{\geq 0}$: Numbers of degree-one and degree-2τ building blocks, respectively.

– $\boldsymbol{\mu} \in \{\, 1, 2 \,\}^{\beta_1+\beta_2}$: Vector determining the order in which the building blocks are multiplied. This vector has exactly β_i i's for both $i = 1, 2$.

– $\mathbf{w} \in \mathbb{F}^{\beta_1}$: Parameters of the $\mathbf{B}_1$ building blocks

– $\boldsymbol{\zeta} \in \mathbb{F}^{\beta_2}$, $\boldsymbol{\tau} \in \mathbb{N}^{\beta_2}$: Parameters of the $\mathbf{S}$ building blocks

We note that a 2×2 degree-one building block is completely determined with a single finite-field element. To see why, we recall that the vector $\mathbf{v} \in \mathbb{F}^2$ in the definition of this building block must have unit norm. Since the characteristic of the underlying field is two, it can be easily verified that every length-two unit-norm vector is of the form $[\, v \quad 1+v \,]^\dagger$ for an arbitrary $v \in \mathbb{F}$. Therefore, in the parameters listed for the construction of univariate 2×2 PU matrices, the vector $\mathbf{w}$ suffices to specify all degree-one building blocks. Based on this description, Algorithm 2.2 can be used to generate univariate 2×2 PU matrices.

Algorithm 2.2: 2×2 PU matrix generation.

INPUT: Integers $\beta_1, \beta_2 \in \mathbb{Z}_{\geq 0}$ and vectors $\boldsymbol{\mu} \in \{\, 1, 2 \,\}^{\beta_1+\beta_2}$, $\mathbf{w} \in \mathbb{F}^{\beta_1}$, $\boldsymbol{\zeta} \in \mathbb{F}^{\beta_2}$, and $\boldsymbol{\tau} \in \mathbb{N}^{\beta_2}$

OUTPUT: A PU matrix polynomial $\mathbf{P}(z) \in \mathsf{PU}_2(\mathbb{F}[z^{-1}])$

▷ Components of an arbitrary vector $\mathbf{x}$ are denoted by $x_1, x_2, \ldots$.

1. $\mathbf{P}(z) \leftarrow \mathbf{I}$, $i, j \leftarrow 0$
2. **for** $\ell = 1$ **to** $\beta_1 + \beta_2$ **do**
3. **case** $\mu_\ell = 1$, $i \leftarrow i + 1$, $\mathbf{P}(z) \leftarrow \mathbf{P}(z)\, \mathbf{B}_1(z; [\, w_i, \; 1 + w_i \,]^\dagger)$
4. **case** $\mu_\ell = 2$, $j \leftarrow j + 1$, $\mathbf{P}(z) \leftarrow \mathbf{P}(z)\, \mathbf{S}_{2\tau_j}(z; \zeta_j)$
5. **end**
6. **return** $\mathbf{P}(z)$

CHAPTER 3

Factorization of Bivariate PU Matrices

Motivated by the applications of multivariate PU matrix polynomials in cryptography [60, 63, 33] and error-control coding [86, 171, 172], we study the generation of bivariate PU matrices through multiplying a finite set of fully-parameterized PU building blocks. The factorization proposed in this chapter is a multilevel technique [56, 65]. In this method, a bivariate PU matrix polynomial is considered a univariate matrix polynomial in one of the variables that its coefficients are polynomial matrices in the other variable. Precisely, an arbitrary bivariate matrix polynomial $\mathbf{P}(x,y) \in \mathsf{M}_M(\mathbb{F}[x^{-1},y^{-1}])$, for any $M \in \mathbb{N}$, can always be formulated as

$$\mathbf{P}(x,y) = \sum_{i=0}^{L} \mathbf{P}_i(x)\, y^{-i} \quad , \tag{3.1}$$

where $L \in \mathbb{Z}_{\geq 0}$ is the order with respect to y^{-1} and $\mathbf{P}_i(x) \in \mathsf{M}_M(\mathbb{F}[x^{-1}])$ for all $i = 0,\ldots,L$. The goal is treating $\mathbf{P}(x,y)$ as a univariate matrix polynomial over the ring $\mathbb{F}[x^{-1}]$. Therefore, we extend the definition of PU matrices to the ring of polynomials as follows.

Definition 3.0.1 (PU over the Ring $\mathbb{F}[x^{\pm 1}]$). *A bivariate matrix polynomial $\mathbf{P}(x,y) \in \mathsf{M}_M(\mathbb{F}[x^{\pm 1},y^{-1}])$ is PU over the ring $\mathbb{F}[x^{\pm 1}]$ of Laurent polynomials if and only if*

$$\mathbf{P}^{\dagger}(x,y)\,\mathbf{P}(x,y) = \overline{\alpha}(x)\,\alpha(x)\,\mathbf{I} \tag{3.2}$$

in which $\alpha(x) \in \mathbb{F}[x^{\pm 1}] \setminus \{\,0\,\}$ is an arbitrary nonzero polynomial.

This generalization is in line with a general definition of PU matrices over the complex field $\mathbb{C}$. In this general form, a matrix polynomial $\mathbf{P}(x,y) \in \mathsf{M}_M(\mathbb{C}[x^{-1},y^{-1}])$ is called PU if and only if $\mathbf{P}^{\dagger}(x,y)\,\mathbf{P}(x,y) = \overline{c}\,c\,\mathbf{I}$ for some $c \in \mathbb{C}^{\times}$, where $\overline{c}$ is the complex conjugate of c.

Using the representation in (3.1), we provide a complete factorization for $\mathbf{P}(x, y)$ in terms of bivariate building blocks that are PU over $\mathbb{F}[x^{\pm 1}]$. This gives a first-level factorization. The possibility of a second-level factorization depends on the terms obtained in the first level.

The organization of this chapter is as follows. In Section 3.1, we briefly review the related work in the factorization of multivariate PU matrix polynomials. Some mathematical results regarding the structure of self-orthogonal vector polynomials are gathered in Section 3.2. These results are used in the following sections. The bivariate degree-one and degree-2τ building blocks are introduced in Section 3.3. Using these building blocks, a first-level factorization is provided in Section 3.4. In Section 3.5, the cases in which a second-level factorization are possible are studied. The application of the proposed multilevel factorization technique to the factorization of multivariate PU matrix polynomials over the complex field and also the design of two-dimensional burst error-correcting codes is briefly studied in Section 3.6.

3.1 Related Work

The problem of factorizing unitary and univariate PU matrices over the Galois field $\mathsf{GF}(2)$ was first addressed in [161, 160]. In these works, the univariate degree-one and degree-two PU building blocks (defined in (2.9) and (2.10), respectively) were introduced. It was shown that these building blocks cannot generate all univariate PU matrices over $\mathsf{GF}(2)$. However, a complete factorization for the subclass of lapped orthogonal transform (LOT) was presented. This subclass consists of univariate PU matrices of the form $\mathbf{p}_0 + \mathbf{p}_1 x^{-1}$ in which $\mathbf{p}_0, \mathbf{p}_1 \in \mathsf{M}_M(\mathbb{F})$. LOTs have applications in audio and image processing [35, 36] and also watermarking [129]. State-space representation of univariate PU matrices over $\mathsf{GF}(2)$ and a factorization of PU matrices over $\mathsf{GF}(q)$ for $q > 2$ are also examined in [161, 160].

The factorization of unitary matrices and univariate two-channel PU filter banks over $\mathsf{GF}(2^r)$, for an arbitrary $r \in \mathbb{N}$, is studied in [86]. To complete the factorization, the degree-

2τ building block (defined in (2.11)) is introduced. For the factorization of univariate $M \times M$ PU matrices with $M \geq 3$, a new building block is also presented. It is conjectured that this building block along with the degree-one and degree-two building blocks of [161] complete the factorization.

Most of the work in the factorization of multivariate filter banks has been done over real or complex field [186, 190]. Unfortunately, the results cannot be easily carried over to filter banks over finite fields. Kronecker product of univariate PU matrices to generate bivariate systems is proposed in [51]. In this method, if $\mathbf{A}(x)$ and $\mathbf{B}(y)$ are two univariate PU matrix polynomials (not necessarily of the same size), then a bivariate PU matrix polynomial is generated as $\mathbf{A}(x) \otimes \mathbf{B}(y)$, where $\otimes$ is the Kronecker product operator. Let $\mathbf{A}(x) = [\, a_{i,j}(x) \,]$ be an $M \times M$ PU matrix polynomials. Then, the Kronecker product $\mathbf{A}(x) \otimes \mathbf{B}(y)$ is defined as

$$\mathbf{A}(x) \otimes \mathbf{B}(y) := \begin{bmatrix} a_{1,1}(x)\,\mathbf{B}(y) & \cdots & a_{1,M}(x)\,\mathbf{B}(y) \\ \vdots & \ddots & \vdots \\ a_{M,1}(x)\,\mathbf{B}(y) & \cdots & a_{M,M}(x)\,\mathbf{B}(y) \end{bmatrix} . \tag{3.3}$$

This method does not generate all bivariate PU matrices even over the real field.

It is proved in [145] that bivariate PU matrices admit a factorization consisting of the product of FIR PU factors. However, those factors are neither determined nor parameterized. Construction of bivariate PU matrices through multiplying univariate degree-one building blocks with respect to any one of the two variables in an arbitrary order is studied in [113]. In this method, if $\mathbf{C}(x;\, \mathcal{P})$ is a univariate PU building block with the parameter set $\mathcal{P}$, then a bivariate PU matrix polynomial is generated as $\prod_{i=1}^{N} \mathbf{C}(x;\, \mathcal{P}_i)\, \mathbf{C}(y;\, \mathcal{Q}_i)$. It is also conjectured that $\mathbf{C}(x;\, \mathcal{P})\, \mathbf{C}(y;\, \mathcal{Q})$ is a building block for generating all bivariate PU systems. As discussed in [188], such a building block does not give a general factorization.

The importance of Gröbner bases in multidimensional multirate systems is discussed in [111]. It is explained in that paper how Gröbner bases can be used to design multidimensional filter banks, but a general factorization and realization technique is not provided. A family of bivariate nonseparable PU matrices is introduced in [151] without giving any fac-

torization technique. In addition, the non-separability condition of bivariate PU matrices is discussed in this paper

As an alternative approach, Smith-form decomposition is studied in [191, 88]. Using this technique, a polynomial matrix can be written as a product of a diagonal matrix packed in between two unimodular matrices. The disadvantage of this technique is that unimodular matrices cannot be well parameterized.

The factorization of a subclass of bivariate matrix polynomials in $\mathsf{PU}_M(\mathbb{C}[x^{-1}, y^{-1}])$ is studied in [128]. This subclass consists of bivariate matrix polynomials, such as $\mathbf{P}(x, y)$, that have order one with respect to y^{-1}.

$$\mathbf{P}(x, y) = \sum_{i=0}^{L} \sum_{j=0}^{1} \mathbf{p}_{i,j}\, x^{-i}\, y^{-j} \tag{3.4}$$

The core of the factorization is the following lemma.

LEMMA 3.1.1. *Consider a matrix polynomial* $\mathbf{P}(x, y) \in \mathsf{PU}_M(\mathcal{F}[x^{-1}, y^{-1}])$ *with* $\mathsf{Ord}_{y^{-1}}\mathbf{P} = 1$ *as in (3.4), where* $\mathcal{F}$ *is an arbitrary field. Assume that the coefficients* $\mathbf{p}_{0,0}$ *and* $\mathbf{p}_{0,1}$ *are noth both zero. Similarly, assume the coefficients* $\mathbf{p}_{L,0}$ *and* $\mathbf{p}_{L,1}$ *are not both zero. There exists a nonzero constant vector* $\mathbf{v} \in \mathcal{F}^M$ *such that either*

$$\mathbf{v}^\dagger \mathbf{p}_{0,0} = \mathbf{v}^\dagger \mathbf{p}_{0,1} = \mathbf{0} \tag{3.5a}$$

or

$$\mathbf{p}_{0,0}\mathbf{v} = \mathbf{p}_{0,1}\mathbf{v} = \mathbf{0} \ . \tag{3.5b}$$

This lemma is proved in [128] for the case $\mathcal{F} = \mathbb{C}$. However, we claim that it holds in general. For completeness and also to prove this claim, we have provided the proof in Appendix A. An immediate result of Lemma 3.1.1 is the following corollary.

COROLLARY 3.1.1. *Consider a matrix polynomial* $\mathbf{P}(x, y) \in \mathsf{PU}_M(\mathcal{F}[x^{-1}, y^{-1}])$ *with the property* $\mathsf{Ord}_{y^{-1}}\mathbf{P} = 1$ *as in (3.4). Let* $\mathbf{P}_0(y) := \mathbf{p}_{0,0} + \mathbf{p}_{0,1}y^{-1}$. *There exists a nonzero vector* $\mathbf{v} \in \mathcal{F}^M$ *such that either* $\mathbf{v}^\dagger \mathbf{P}_0(y) = \mathbf{0}$ *or* $\mathbf{P}_0(y)\mathbf{v} = \mathbf{0}$.

3.2 Self-Orthogonal Polynomial Vectors over $\mathbb{F}[x^{\pm 1}]$

As explained in the previous section, our approach in the factorization of bivariate PU matrices is providing a two-level factorization. In the first level, the bivariate PU polynomial matrix is considered a univariate one with its coefficients being polynomials in the other variable. To find building blocks for this level of factorization, in this section, we study the structure of self-orthogonal polynomial vectors in the module $(\mathbb{F}[x^{\pm 1}])^2$.

Vector spaces over the complex field and modules over the ring of polynomials behave very similarly, but there are some major differences. One difference is the presence of self-orthogonal polynomial vectors in the module $(\mathbb{F}[x^{\pm 1}])^2$. A polynomial vector $\mathbf{v}(x) \in (\mathbb{F}[x^{\pm 1}])^2$ is self-orthogonal if and only if $\langle \mathbf{v}(x), \mathbf{v}(x) \rangle = 0$, where the sesquilinear form $\langle \cdot, \cdot \rangle$ is defined in (1.6). For example, the polynomial vector

$$\mathbf{v}(x) = \begin{bmatrix} x^m p(x) \\ p(x^{-1}) \end{bmatrix} \quad \in \ (\mathbb{F}[x^{\pm 1}])^2 \tag{3.6}$$

is self-orthogonal since[1]

$$\begin{aligned} \langle \mathbf{v}(x), \mathbf{v}(x) \rangle &= p(x)p(x^{-1}) + p(x)p(x^{-1}) \\ &= 0 \ . \end{aligned}$$

In fact, by the following lemma, every self-orthogonal polynomial vector is of this form.

LEMMA 3.2.1. *Every self-orthogonal polynomial vector in the module* $(\mathbb{F}[x^{\pm 1}])^2$ *has the form*

$$\beta(x) \begin{bmatrix} x^m p(x) \\ p(x^{-1}) \end{bmatrix} \ ,$$

where $\beta(x), p(x) \in \mathbb{F}[x^{\pm 1}]$ *and* $m \in \mathbb{Z}$ *are arbitrary elements.*

The proof of this lemma is given in Appendix A. Polynomial vectors orthogonal to self-orthogonal polynomial vectors also have a special form specified by the following lemma.

[1] We note that the characteristic of the underlying field is two.

LEMMA 3.2.2. *Consider the self-orthogonal vector* $\mathbf{v}(x) = \beta(x)\,\mathbf{v}'(x)$*, where* $\beta(x) \in \mathbb{F}[x^{\pm 1}]$ *and* $\mathbf{v}'(x)$ *is as in (3.6). In addition, suppose* $x^m p(x)$ *and* $p(x^{-1})$ *do not have any common factor. Then, every vector orthogonal to* $\mathbf{v}(x)$ *has the form* $\beta'(x)\,\mathbf{v}'(x)$ *for some* $\beta'(x) \in \mathbb{F}[x^{\pm 1}]$.

The proof is given in Appendix A. Using this lemma, we can easily prove the following fact about rank-deficient polynomial matrices.

FACT 3.2.1. *Any rank-deficient polynomial matrix* $\mathbf{A}(x)$ *in the algebra* $\mathsf{M}_2(\mathbb{F}[x^{\pm 1}])$ *that its left null-space is self-dual has the form*

$$\mathbf{A}(x) = \beta(x) \begin{bmatrix} x^m p(x) \\ p(x^{-1}) \end{bmatrix} \mathbf{w}^{\dagger}(x) \ ,$$

where $m \in \mathbb{Z}$*,* $\beta(x), p(x) \in \mathbb{F}[x^{\pm 1}]$*, and* $\mathbf{w}(x) \in (\mathbb{F}[x^{\pm 1}])^2$ *are arbitrary elements.*

A similar fact can be stated about rank-deficient matrices with their right null-spaces being self-dual. The combination is the following fact.

FACT 3.2.2. *Any rank-deficient polynomial matrix* $\mathbf{A}(x)$ *in the algebra* $\mathsf{M}_2(\mathbb{F}[x^{\pm 1}])$ *that its left and right null-spaces are both self-dual has the form*

$$\mathbf{A}(x) = \beta(x) \begin{bmatrix} x^m p(x) \\ p(x^{-1}) \end{bmatrix} \begin{bmatrix} x^{-r} q(x^{-1}) & q(x) \end{bmatrix} \ ,$$

where $m, r \in \mathbb{Z}$ *and* $\beta(x), p(x), q(x) \in \mathbb{F}[x^{\pm 1}]$ *are arbitrary elements.*

Using the background given in this section, we introduce elementary building blocks over the ring $\mathbb{F}[x^{\pm 1}]$ for the first-level factorization of bivariate PU matrices.

3.3 Bivariate Building Blocks PU over $\mathbb{F}[x^{\pm 1}]$

Similar to the univariate case, it is desirable to have parameterized building blocks to generate all bivariate PU matrices. In this section, we provide two elementary building blocks for the first-level factorization of matrices in $\mathsf{PU}_2(\mathbb{F}[x, y])$. These building blocks are

defined over the ring $\mathbb{F}[x^{\pm 1}]$ of Laurent polynomials. We will prove in the next section that these building blocks are sufficient to provide a first-level factorization for all bivariate PU matrices.

3.3.1 Bivariate Degree-One PU Building Block

The generalization of the degree-one building block, defined in (2.9), is

$$\mathsf{B}_1\Big(y;\, \mathbf{v}(x)\Big) := \alpha(x)\,\mathbf{I} + \mathbf{v}(x)\,\mathbf{v}^{\dagger}(x) + \mathbf{v}(x)\,\mathbf{v}^{\dagger}(x)\,y^{-1} \tag{3.7}$$

in which $\mathbf{v}(x) \in (\mathbb{F}[x^{\pm 1}])^2$ is a polynomial vector such that $\mathbf{v}^{\dagger}(x)\,\mathbf{v}(x) =: \alpha(x) \in \mathbb{F}[x^{\pm 1}] \setminus \{\,0\,\}$. By its definition, $\alpha(x)$ is a symmetric polynomial, i.e., $\overline{\alpha}(x) = \alpha(x)$. By direct computation, one can show that $\mathsf{B}_1\Big(y;\, \mathbf{v}(x)\Big)$ is PU over the ring $\mathbb{F}[x^{\pm 1}]$, i.e.,

$$\mathsf{B}_1^{\dagger}\Big(y;\, \mathbf{v}(x)\Big)\,\mathsf{B}_1\Big(y;\, \mathbf{v}(x)\Big) = \alpha^2(x)\,\mathbf{I}\ . \tag{3.8}$$

Similar to the univariate case, the determinant of a bivariate PU matrix gives its degree with respect to its variables. Since the introduced building block is PU over a ring, its determinant gives its degree only with respect to y. The following lemma provides the value of the determinant.

LEMMA 3.3.1. *The determinant of the building block defined in (3.7) is*

$$\det \mathsf{B}_1\Big(y;\, \mathbf{v}(x)\Big) = \alpha^2(x)\,y^{-1}\ .$$

Proof. It can be easily shown that $\mathsf{B}_1\Big(y;\, \mathbf{v}(x)\Big)\,\mathbf{v}(x) = \alpha(x)\,y^{-1}\,\mathbf{v}(x)$ that implies $\mathbf{v}(x)$ is an eigenvector with the eigenvalue $\alpha(x)\,y^{-1}$. Furthermore, if $\mathbf{u}(x) \in (\mathbb{F}[x^{\pm 1}])^2$ is orthogonal to $\mathbf{v}(x)$, we have $\mathsf{B}_1\Big(y;\, \mathbf{v}(x)\Big)\,\mathbf{u}(x) = \alpha(x)\,\mathbf{u}(x)$, which has a similar implication. One can easily show that every length-two polynomial vector has an orthogonal polynomial vector in $(\mathbb{F}[x^{\pm 1}])^2$. Since the matrix under study is 2×2, it has at most two distinct eigenvectors, which we have determined them. The proof is complete by noting that the determinant of an arbitrary matrix equals the product of its eigenvalues. ∎

By Lemma 3.3.1, the degree of the building block $\mathbf{B}_1\Big(y;\, \mathbf{v}(x)\Big)$ with respect to y is one. This result is also verified by the block diagram of this building block shown in Figure 3.1. Two useful observations are stated in the following facts.

FACT 3.3.1. *The polynomial matrix* $\mathbf{B}_1^{\dagger}\Big(y;\, \mathbf{v}(x)\Big)$ *is PU over the ring* $\mathbb{F}[x^{\pm 1}]$ *as well, i.e.,*

$$\mathbf{B}_1\Big(y;\, \mathbf{v}(x)\Big)\, \mathbf{B}_1^{\dagger}\Big(y;\, \mathbf{v}(x)\Big) = \alpha^2(x)\, \mathbf{I} \ .$$

FACT 3.3.2. *The polynomial* $\alpha(x)$ *in (3.7) is a* normalizer *since* $\frac{1}{\alpha(x)}\, \mathbf{B}_1\Big(y;\, \mathbf{v}(x)\Big)$ *is PU over* $\mathbb{F}$.

3.3.2 Bivariate Degree-2τ PU Building Block

To continue the first-level factorization when encountering a self-orthogonal polynomial vector, we introduce the bivariate degree-2τ PU building block as follows.

$$\mathbf{S}_{2\tau}\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big) := \zeta(x)\, \mathbf{u}(x)\, \mathbf{v}^{\dagger}(x) + \begin{bmatrix} x^{m-r}\, p(x)\, q(x^{-1}) & 0 \\ 0 & p(x^{-1})\, q(x) \end{bmatrix} y^{-\tau} + \zeta(x)\, \mathbf{u}(x)\, \mathbf{v}^{\dagger}(x)\, y^{-2\tau} \tag{3.9}$$

Here, $\tau \in \mathbb{N}$, $\zeta(x) \in \mathbb{F}[x^{\pm 1}]$ is a symmetric polynomial (i.e., $\overline{\zeta}(x) = \zeta(x)$), and

$$\mathbf{u}(x) = \begin{bmatrix} x^m\, p(x) \\ p(x^{-1}) \end{bmatrix} \in \mathbb{F}[x^{\pm 1}] \ , \qquad \mathbf{v}(x) = \begin{bmatrix} x^r\, q(x) \\ q(x^{-1}) \end{bmatrix} \in \mathbb{F}[x^{\pm 1}] \ . \tag{3.10}$$

By the factorization algorithm (presented in the next section), the introduced building block cannot be factored using the degree-one building block in (3.7) since both vectors in (3.10)

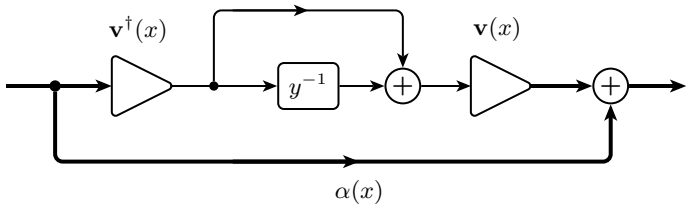


Figure 3.1: Block diagram of the bivariate degree-one PU building block.

are self-orthogonal. By a direct calculation, we can easily show that the building block introduced in (3.9) is PU over $\mathbb{F}[x^{\pm 1}]$, i.e.,

$$\mathsf{S}_{2\tau}^{\dagger}\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big)\, \mathsf{S}_{2\tau}^{\dagger}\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big) = p(x)\, p(x^{-1})\, q(x)\, q(x^{-1})\, \mathbf{I} \ . \tag{3.11}$$

To determine the degree of S with respect to y, we directly calculate its determinant.

FACT 3.3.3. *The determinant of the building block defined in (3.9) is*

$$\det \mathsf{S}_{2\tau}\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big) = x^{m-r}\, p(x)\, p(x^{-1})\, q(x)\, q(x^{-1})\, y^{-2\tau} \ .$$

By this fact, the introduced building block has degree 2τ with respect to y. This fact is verified by the block diagram in Figure 3.2 in which the polynomial vectors $\mathbf{u}'(x)$ and $\mathbf{v}'(x)$ are

$$\mathbf{u}'(x) = \begin{bmatrix} x^m\, p(x) \\ 0 \end{bmatrix} \ , \qquad \mathbf{v}'(x) = \begin{bmatrix} 0 \\ q(x^{-1}) \end{bmatrix} \ .$$

Similar to the degree-one building block, the following two facts will be used later in this chapter.

FACT 3.3.4. *The polynomial matrix* $\mathsf{S}_{2\tau}^{\dagger}\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big)$ *is PU over the ring* $\mathbb{F}[x^{\pm 1}]$ *as well, i.e.,*

$$\mathsf{S}_{2\tau}\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big)\, \mathsf{S}_{2\tau}^{\dagger}\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big) = p(x)\, p(x^{-1})\, q(x)\, q(x^{-1})\, \mathbf{I} \ .$$

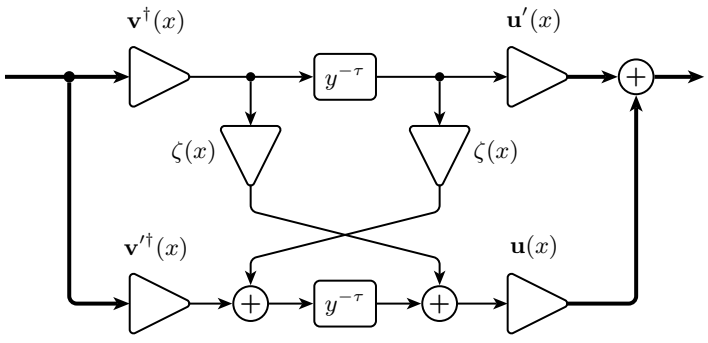


Figure 3.2: Block diagram of the bivariate degree-2τ PU building block.

FACT 3.3.5. *The polynomial* $p(x)\,q(x)$ *in (3.9) is a* normalizer *since*

$$\frac{1}{p(x)\,q(x)}\,\mathsf{S}_{2\tau}\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big)$$

is PU over $\mathbb{F}$.

Using the building blocks introduced in this section, we introduce the first-level factorization algorithm over the ring $\mathbb{F}[x^{\pm 1}]$.

3.4 First-Level Factorization over $\mathbb{F}[x^{\pm 1}]$

In this section, using the building blocks introduced in the previous section, we present a complete first-level factorization for all bivariate polynomial matrices in $\mathsf{PU}_2(\mathbb{F}[x^{-1},y^{-1}])$. The first step of factorization is to reformulate a given bivariate PU matrix as a matrix polynomial in y^{-1} that its coefficients are matrices in $\mathsf{M}_2(\mathbb{F}[x^{-1}])$. Using this representation, we can always extract one of the bivariate building blocks from either the left or the right of the PU matrix. This extraction reduces the degree of the PU matrix with respect to y. Repeating this process a finite number of times, we eventually obtain a polynomial matrix independent of y that is PU over $\mathbb{F}[x^{\pm 1}]$. This terminates the factorization algorithm.

Consider an arbitrary matrix $\mathbf{P}(x,y) \in \mathsf{PU}_2(\mathbb{F}[x^{-1},y^{-1}])$. As explained at the beginning of this chapter, it can always be reformulated as

$$\mathbf{P}(x,y) = \sum_{i=1}^{L} \mathbf{P}_i(x)\,y^{-i} \;\;, \tag{3.12}$$

where $L = \mathsf{Ord}_{y^{-1}}\mathbf{P}$ and $\mathbf{P}_i(x) \in \mathsf{M}_2(\mathbb{F}[x^{-1}])$ for all $i = 0,1,\ldots,L$. Without loss of generality, we assume $\mathbf{P}_0(x) \neq \mathbf{0}$. Otherwise, we factor out y^{-1} and continue the factorization with the remainder. By the PU property of $\mathbf{P}(x,y)$, we have

$$\left[\sum_{i=1}^{L} \mathbf{P}^{\dagger}(x)\,y^{i}\right]\;\left[\sum_{j=1}^{L} \mathbf{P}(x)\,y^{-j}\right] = \mathbf{I}$$

from which we conclude that the coefficient of y^L must be zero, i.e.,

$$\mathbf{P}_L^{\dagger}(x)\,\mathbf{P}_0(x) = \mathbf{0} \;\;. \tag{3.13}$$

Since $\mathbf{P}_0(x) \neq \mathbf{0}$ by our assumption, this result indicates that $\mathbf{P}_0(x)$ is singular and all column vectors of $\mathbf{P}_L(x)$ are in $\mathsf{Null}_L(\mathbf{P}_0(x))$. Since $\mathbf{P}(x,y)$ is a square matrix, $\mathbf{P}^\dagger(x,y)$ is PU as well from where we obtain

$$\mathbf{P}_0(x)\,\mathbf{P}_L^\dagger(x) = \mathbf{0} \ . \tag{3.14}$$

Similarly, this result indicates that all column vectors of $\mathbf{P}_L^\dagger(x)$ are in $\mathsf{Null}_R(\mathbf{P}_0(x))$.

The factorization algorithm is similar to the univariate case. We first attempt to extract the degree-one building blocks from either the right or the left of the PU matrix. The extraction of such factors fails only when both the left and the right null spaces of the matrix coefficient of y^0 (in a polynomial expansion similar to (3.12)) are self-dual. In this situation, the factorization is continued using the degree-2τ building block. The goal is proving that these two building blocks suffice to completely factorize all bivariate PU matrices. The following lemma describes the situation in which a degree-one building block can be extracted.

LEMMA 3.4.1. *Suppose* $\mathsf{Null}_L(\mathbf{P}_0(x))$ *is not self-dual. Randomly pick a polynomial vector* $\mathbf{v}(x) \in \mathsf{Null}_L(\mathbf{P}_0(x))$ *such that* $\alpha(x) = \mathbf{v}^\dagger(x)\,\mathbf{v}(x) \neq 0$. *The matrix polynomial* $\mathbf{P}(x,y)$ *can be factored as*

$$\alpha^2(x)\,\mathbf{P}(x,y) = \mathbf{B}_1\Big(y;\,\mathbf{v}(x)\Big)\,\mathbf{P}'(x,y) \ ,$$

where $\mathbf{P}'(x,y) \in \mathsf{M}_2(\mathbb{F}[x^{\pm 1}, y^{-1}])$ *is PU over* $\mathbb{F}[x^{\pm 1}]$ *such that* $\mathsf{Deg}_y\mathbf{P}' = \mathsf{Deg}_y\mathbf{P} - 1$.

The proof of this lemma is provided in Appendix A. A similar lemma can be stated about the extraction of a degree-one building block from the right.

LEMMA 3.4.2. *Suppose* $\mathsf{Null}_R(\mathbf{P}_0(x))$ *is not self-dual. Randomly pick a polynomial vector* $\mathbf{v}(x) \in \mathsf{Null}_R(\mathbf{P}_0(x))$ *among the column vectors of* $\mathbf{P}_L(x)$ *such that* $\alpha(x) = \mathbf{v}^\dagger(x)\,\mathbf{v}(x) \neq 0$. *The matrix polynomial* $\mathbf{P}(x,y)$ *can be factored as*

$$\alpha^2(x)\,\mathbf{P}(x,y) = \mathbf{P}'(x,y)\,\mathbf{B}_1\Big(y;\,\mathbf{v}(x)\Big) \ ,$$

where $\mathbf{P}'(x,y) \in \mathsf{M}_2(\mathbb{F}[x^{\pm 1}, y^{-1}])$ *is PU over* $\mathbb{F}[x^{\pm 1}]$ *such that* $\mathsf{Deg}_y\mathbf{P}' = \mathsf{Deg}_y\mathbf{P} - 1$.

By the previous two lemmas, the extraction of degree-one building block fails when both null spaces of $\mathbf{P}_0(x)$ are self-dual. In the following theorem, we prove that the degree-2τ building block can be extracted in this situation.

THEOREM 3.4.1. *Assume both null spaces of $\mathbf{P}_0(x)$ in (3.12) are self-dual. There exist $\tau \in \mathbb{N}$ and self-orthogonal polynomial vectors $\mathbf{u}(x), \mathbf{v}(x) \in (\mathbb{F}[x^{\pm 1}])^2$ such that $\mathbf{P}(x,y)$ can be factored as either*

$$p(x)q(x)\mathbf{P}(x,y) = y^{\tau-1}\,\mathbf{S}_{2\tau}\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big)\,\mathbf{P}'(x,y) \tag{3.15a}$$

or

$$p(x)q(x)\mathbf{P}(x,y) = y^{\tau-1}\,\mathbf{P}'(x,y)\,\mathbf{S}_{2\tau}\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big)\ . \tag{3.15b}$$

In these equations, the polynomial matrix $\mathbf{P}'(x,y) \in \mathsf{M}_2(\mathbb{F}[x^{\pm 1}, y^{-1}])$ is PU over $\mathbb{F}[x^{\pm 1}]$ and $\mathsf{Deg}_y\mathbf{P}' = \mathsf{Deg}_y\mathbf{P} - 2\tau$.

The proof is provided in Appendix A. The results obtained so far pave the way for providing a complete first-level factorization as summarized in the following theorem.

THEOREM 3.4.2 (First-Level Factorization). *Let $\mathbf{P}(x,y) \in \mathsf{PU}_2(\mathbb{F}[x^{-1}, y^{-1}])$. This polynomial matrix can always be factored as*

$$\begin{aligned}\mathbf{P}(x,y) = y^t\ &\left[\frac{1}{\eta_N(x)}\mathbf{C}_N(x,y)\right]\cdots\left[\frac{1}{\eta_{N'+1}(x)}\mathbf{C}_{N'+1}(x,y)\right]\ \left[\frac{1}{\prod_{i=1}^{N}\eta_i(x)}\mathbf{F}(x)\right]\\ &\times\left[\frac{1}{\eta_{N'}(x)}\mathbf{C}_{N'}(x,y)\right]\cdots\left[\frac{1}{\eta_1(x)}\mathbf{C}_1(x,y)\right]\ .\end{aligned} \tag{3.16}$$

Here, $t \in \mathbb{Z}$, $N \le \mathsf{Deg}_y\mathbf{P}$, $N' \in [\,N\,]$, $\mathbf{C}_i(x,y)$ is one of the two bivariate building blocks in (3.7) and (3.9), and $\eta_i(x) \in \mathbb{F}[x^{\pm 1}]$ is the corresponding normalizer as in Fact 3.3.2 and Fact 3.3.5 for all $i \in [\,N\,]$.

Proof. To prove this theorem, we provide an algorithm that gives the desired factorization. By (3.13) and (3.14), $\mathbf{P}_0(x)$ is rank deficit. Therefore, there is either a nonzero polynomial vector $\mathbf{u}(x) \in \mathsf{Null}_L(\mathbf{P}_0(x))$ or a nonzero vector $\mathbf{v}(x) \in \mathsf{Null}_R(\mathbf{P}_0(x))$. If at least one of these two vectors is non self-orthogonal, we can extract a degree-one building block by either

Lemma 3.4.1 or Lemma 3.4.2. Otherwise, there exists some $\tau \in \mathbb{N}$ that we can factor a degree-2τ building block by Theorem 3.4.1. In either case, the result is a polynomial matrix in $\mathsf{M}_2(\mathbb{F}[x^{\pm 1}, y^{-1}])$ that is PU over $\mathbb{F}[x^{\pm 1}]$, and its degree with respect to y is reduced. We emphasize that to apply any one of the algorithms for the extraction of a bivariate degree-one or degree-2τ building block, the input polynomial matrix is sufficient to be PU over $\mathbb{F}[x^{\pm 1}]$. This algorithm terminates after at most $\mathsf{Deg}_y\mathbf{P}$ steps after which we obtain a polynomial matrix with degree zero with respect to y. This implies that the reminder is independent of y; otherwise, another building block can be extracted. After the completion of the factorization, some simple mathematical manipulations give a factorization as in (3.16). ∎

Terms in the factorization provided by Theorem 3.4.2 belong to the set $\mathsf{M}_2(\mathbb{F}(x^{\pm 1})[y^{-1}])$, bivariate matrices that are rational in x, but polynomial in y. If for some i, the normalizer $\eta_i(x)$ is a nonzero constant in $\mathbb{F}$, then the term $\frac{1}{\eta_i(x)}\mathbf{C}_i(x, y)$ belongs to $\mathsf{M}_2(\mathbb{F}[x^{\pm 1}, y^{-1}])$. In the next section, we show how to further factorize such factors. The factorization in (3.16) becomes a polynomial in both variables if and only if $\eta_i(x) \in \mathbb{F}^\times$ for all $i \in [\,N\,]$. In this case, the term $\frac{1}{\prod_{i=1}^{N}\eta_i(x)}\mathbf{F}(x)$ becomes PU over $\mathbb{F}$. Since it is univariate, it can be factorized using the univariate PU building blocks and delays in x if necessary.

The existence of rational factors in (3.16) might seem strange since such terms do not appear in the factorization of univariate PU matrices. The major difference between the univariate factorization, studied in [186], and its bivariate version in (3.16) is that the former is performed over the real or complex field whereas the later is done over the ring $\mathbb{F}[x^{\pm 1}]$. Since every nonzero element in a ring does not necessarily have a multiplicative inverse, we get fractions in (3.16). The technique employed here is, in a sense, similar to "lifting" used in algebraic geometry and category theory. The core of this technique is considering a problem in an algebraic structure higher than the one it is already defined in. Using the tools available in the higher structure, one may easily solve the problem. The final answer is the projection of the solution to the lower algebraic structure. For example, consider the factorization of univariate polynomials over a field. This factorization is always

possible over the algebraic closure of the base field in which the polynomial splits into linear factors [122]. However, the coefficients of these factors lie in the algebraic closure. To get polynomial factors over the base field, it might be necessary to multiply some of the linear factors.

In the factorization of an arbitrary polynomial matrix $\mathbf{P}(x,y) \in \mathsf{PU}_2(\mathbb{F}[x^{-1},y^{-1}])$, we have a similar problem. The objective is to make use of the technique developed for the factorization of univariate PU matrix polynomials. Hence, the goal is to consider $\mathbf{P}(x,y)$ as a univariate polynomial matrix in y that its coefficients are themselves Laurent matrix polynomials in x. To this end, we take several steps. Lifting to the ring of Laurent polynomials allows taking advantage of the properties of such polynomials namely the extended definition of PU matrices over the ring of polynomials. In the first step, we use the proper inclusion $\mathbb{F}[x^{-1}] \subsetneq \mathbb{F}[x^{\pm 1}]$ to consider $\mathbf{P}(x,y)$ as a matrix polynomial in $\mathbb{F}[x^{\pm 1},y^{-1}]$. To treat $\mathbf{P}(x,y)$ as a univariate matrix polynomial, we use the isomorphism $\mathbb{F}[x^{\pm 1},y^{-1}] \cong \mathbb{F}[x^{\pm 1}][y^{-1}]$. Eventually, in the last step, we lift the factorization problem to the monoid $\mathbb{F}(x^{\pm 1})[y^{-1}]$. This is because, as observed, some terms in the first level of the factorization require polynomial normalizers. To cover such terms, we lift to the field of fractions constructed by Laurent polynomials, i.e., $\mathbb{F}(x^{\pm 1})$. Collectively, we have employed the following hierarchy of algebraic structures.

$$\mathbb{F} \subsetneq \mathbb{F}[x^{-1},y^{-1}] \subsetneq \mathbb{F}[x^{\pm 1},y^{-1}] \cong \mathbb{F}[x^{\pm 1}][y^{-1}] \subsetneq \mathbb{F}(x^{\pm 1})[y^{-1}] \tag{3.17}$$

To convert the rational factors in in (3.16) into polynomial ones, one approach is multiplying appropriate adjacent factors. Nevertheless, it seems difficult to identify potential factors.

3.5 Second-Level Factorization

Each term in the first-level factorization provided by Theorem 3.4.2 can be further factorized if it belongs to $\mathsf{PU}_2(\mathbb{F}[x^{\pm 1},y^{-1}])$. For example, consider an arbitrary factor $\frac{1}{\eta(x)}\mathbf{C}(x,y)$ in which $\eta(x) = \nu \in \mathbb{F}^{\times}$. There always exists the smallest integer $\ell \in \mathbb{Z}_{\geq 0}$ such that

$$\mathbf{A}(x,y) := \frac{x^{-\ell}}{\nu}\mathbf{C}(x,y) \quad \in \mathsf{PU}_2(\mathbb{F}[x^{-1},y^{-1}]) \ . \tag{3.18}$$

In this section, we provide a factorization for $\mathbf{A}(x,y)$ when $\mathbf{C}(x,y)$ is one of the building blocks introduced in Section 3.3.

3.5.1 Degree-One Building Block

When $\mathbf{C}(x,y)$ in (3.18) is the bivariate degree-one building block defined in (3.7), we can further factorize $\mathbf{A}(x,y)$ using Theorem 3.4.2 by changing the roles of x and y. This result is summarized in the following fact.

FACT 3.5.1. *Let* $\mathbf{A}(x,y) = \frac{x^{-\ell}}{\nu}\mathbf{B}_1\Big(y;\, \mathbf{v}(x)\Big) \in \mathsf{PU}_2(\mathbb{F}[x^{-1},y^{-1}])$ *in which* $\nu = \mathbf{v}^\dagger(x)\,\mathbf{v}(x) \in \mathbb{F}^\times$. *This matrix polynomial can be factored as*

$$\begin{aligned}\mathbf{A}(x,y) = x^m \left[\frac{1}{\rho_K(y)}\mathbf{H}_K(x,y)\right]\cdots\left[\frac{1}{\rho_{K'+1}(y)}\mathbf{H}_{K'+1}(x,y)\right]\left[\frac{1}{\prod_{i=1}^K \rho_i(y)}\mathbf{T}(y)\right] \\ \times\left[\frac{1}{\rho_{K'}(y)}\mathbf{H}_{K'}(x,y)\right]\cdots\left[\frac{1}{\rho_1(y)}\mathbf{H}_1(x,y)\right]\ .\end{aligned} \tag{3.19}$$

Here, $m \in \mathbb{Z}$, $K \le \mathsf{Deg}_x\mathbf{A}$, $K' \in [\,K\,]$, $\mathbf{H}_i(x,y)$ *is either the degree-one building block* $\mathbf{B}_1\Big(x;\, \mathbf{w}_i(y)\Big)$ *or the degree-*2τ *building block* $\mathbf{S}_{2\tau}\Big(x;\, \mathbf{t}(y),\mathbf{w}(y),\xi(y)\Big)$, *and* $\rho_i(y)$ *is the corresponding normalizer for all* $i \in [\,K\,]$.

We note that the order of $\mathbf{A}(x,y)$ with respect to y is one. Hence, by Corollary 3.1.1, there exists a nonzero vector $\mathbf{u} \in \mathbb{F}^2$ such that either $\mathbf{u} \in \mathsf{Null}_L(\mathbf{A}_0(y))$ or $\mathbf{u} \in \mathsf{Null}_R(\mathbf{A}_0(y))$. Here, $\mathbf{A}_0(y)$ is the x-independent matrix coefficient in the expansion $\mathbf{A}(x,y) = \sum_{i=0}^M \mathbf{A}_i(y)\,x^{-i}$. The vector $\mathbf{u}$ can be used in the factorization algorithm provided in Theorem 3.4.2 to extract a building block $\mathbf{H}_i(x,y)$ in (3.19). If $\mathbf{u}$ is not self-orthogonal, the resulting building block resides in $\mathsf{M}_2(\mathbb{F}[x^{-1}])$, and thus, PU over $\mathbb{F}$. When all $\rho_i(y)$'s in (3.19) are constants, the factor $\frac{1}{\prod_{i=1}^K \rho_i(y)}\mathbf{T}(y)$ resides in $\mathsf{PU}_2(\mathbb{F}[y^{-1}])$ in which case, it can be further factorized using the univariate building blocks introduced in Section 2.3.

3.5.2 Degree-2τ Building Block

Assume $\mathbf{C}(x,y)$ in (3.18) is the bivariate degree-2τ building block $\mathbf{S}_{2\tau}\Big(y;\, \mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big)$. We give a factorization for $\mathbf{A}(x,y)$ when $\zeta(x) = \zeta \in \mathbb{F}^\times$ is a nonzero constant. The

assumption is that the normalizer (which is $p(x)\,q(x)$ by Fact 3.3.5) is a nonzero constant, i.e., $p(x)\,q(x) = \nu$. Without loss of generality, we may assume $\nu = 1$. This situation is created when $p(x) = x^k$ and $q(x) = x^{-k}$ for some $k \in \mathbb{Z}$. After substituting these results in the definition of the degree-2τ building block in (3.9) and also in (3.18), we get

$$\begin{aligned}\mathbf{A}(x,y) = \zeta \begin{bmatrix} x^{-\ell+m-r} & x^{-\ell+m-2k} \\ x^{-\ell-r} & x^{-\ell-2k} \end{bmatrix} + \begin{bmatrix} x^{-\ell+m-r+2k} & 0 \\ 0 & x^{-\ell-2k} \end{bmatrix} y^{-\tau} \\ + \zeta \begin{bmatrix} x^{-\ell+m-r} & x^{-\ell+m-2k} \\ x^{-\ell-r} & x^{-\ell-2k} \end{bmatrix} y^{-2\tau} \ .\end{aligned} \tag{3.20}$$

The following lemma gives a complete factorization for this matrix polynomial.

LEMMA 3.5.1. *The bivariate PU matrix polynomial in (3.20) can be factorized as*

$$\mathbf{A}(x,y) = \mathbf{B}_1^a(x;\,\boldsymbol{\psi})\,\mathbf{S}_{2\tau}(x;\,\zeta)\,\mathbf{B}_1^b(x;\,\boldsymbol{\varphi})$$

for some $a, b \in \mathbb{N}$ and unit-norm vectors $\boldsymbol{\psi}, \boldsymbol{\varphi} \in \mathbb{F}^2$.

Proof. Depending on the signs and relative values of m, r, and k, different cases may be considered. We give the proof only for the case $m \geq r \geq 0$ and $k = 0$. Proofs of other cases are very similar.

Since ℓ is the smallest nonnegative integer such that $\mathbf{A}(x,y) \in \mathsf{M}_2(\mathbb{F}[x^{-1}, y^{-1}])$. When $m \geq r \geq 0$, we must have $\ell = m$, which gives

$$\mathbf{A}(x,y) = \zeta \begin{bmatrix} x^{-r} & 1 \\ x^{-m-r} & x^{-m} \end{bmatrix} + \begin{bmatrix} x^{-r} & 0 \\ 0 & x^{-m} \end{bmatrix} y^{-\tau} + \zeta \begin{bmatrix} x^{-r} & 1 \\ x^{-m-r} & x^{-m} \end{bmatrix} y^{-2\tau} \ .$$

First, we examine the possibility of extracting the univariate building block $\mathbf{B}_1(x;\,\boldsymbol{\varphi})$, where $\boldsymbol{\varphi} = [\,1 \quad 0\,]^\dagger$, from the right side of $\mathbf{A}(x,y)$. It can be easily verified that r building blocks of this form can be extracted from the right. The remaining matrix polynomial is

$$\begin{aligned}\mathbf{A}_1(x,y) &:= \mathbf{A}(x,y)\ \left[\mathbf{B}_1^\dagger(x;\,\boldsymbol{\varphi})\right]^r \\ &= \zeta \begin{bmatrix} 1 & 1 \\ x^{-m} & x^{-m} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & x^{-m} \end{bmatrix} y^{-\tau} + \zeta \begin{bmatrix} 1 & 1 \\ x^{-m} & x^{-m} \end{bmatrix} y^{-2\tau} \ .\end{aligned}$$

In the next step, we examine the possibility of extracting $\mathbf{B}_1(x;\,\boldsymbol{\phi})$, where $\boldsymbol{\psi} = [\,0 \quad 1\,]^\dagger$, from the left of $\mathbf{A}_1(x,y)$. It is straightforward to show that m building blocks of this form can be extracted. The remainder is

$$\begin{aligned}\mathbf{A}_2(x,y) &:= \Big[\mathbf{B}_1^\dagger(x;\,\boldsymbol{\psi})\Big]^m\,\mathbf{A}_2(x,y)\\ &= \zeta\begin{bmatrix}1 & 1\\ & \\ 1 & 1\end{bmatrix} + \begin{bmatrix}1 & 0\\ & \\ 0 & 1\end{bmatrix} y^{-\tau} + \zeta\begin{bmatrix}1 & 1\\ & \\ 1 & 1\end{bmatrix} y^{-2\tau}\ ,\end{aligned}$$

the degree-2τ building block. Combining these results, we have

$$\mathbf{A}(x,y) = \mathbf{B}_1^m(x;\,\boldsymbol{\psi})\,\mathbf{S}_{2\tau}(x;\,\zeta)\,\mathbf{B}_1^r(x;\,\boldsymbol{\varphi})\ ,$$

which completes the proof. ∎

In practice, we are interested in designing all bivariate PU matrix polynomials in the ring $\mathsf{M}_2(\mathbb{F}[x^{-1},y^{-1}])$ for applications in signal processing, coding [171, 172], and cryptography [54, 60, 63, 62]. The traditional method for designing such matrix polynomials is multiplying univariate primitive building blocks in different variables in an arbitrary order. However, this method certainly does not generate all bivariate PU matrix polynomials. Hence, an important question is whether we are able to capture all such matrix polynomials using the factorization method provided here. Unfortunately, the answer is negative. Nevertheless, using our method, we are able to generate a family of such matrices that is larger than the class of matrices captured by the traditional method. In the following example, using our technique, we generate a bivariate polynomial matrix that is PU over the field $\mathbb{F}$ and cannot be factorized using univariate PU building blocks.

EXAMPLE 3.5.1. *Consider the bivariate polynomial matrix*

$$\mathbf{P}(x,y) = x^{-\ell}\,\mathbf{S}_2\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big)$$

in which $p(x) = x$, $q(x) = x^2$, $\zeta(x) = 1$, $m = -1$, *and* $r = -2$. *The integer* $\ell \in \mathbb{Z}_{\geq 0}$ *is chosen such that* $\mathbf{P}(x,y) \in \mathsf{M}_2(\mathbb{F}[x^{-1},y^{-1}])$. *With this choice of parameters and by*

Fact 3.3.3, one can easily show that $\mathbf{P}(x,y)$ *is PU over* $\mathbb{F}$*. After substituting the parameters and choosing* $\ell = 2$*, we obtain the simplified form*

$$\mathbf{P}(x,y) = \underbrace{\begin{bmatrix} x^{-2} & 1 \\ x^{-3} & x^{-1} \end{bmatrix}}_{\mathbf{P}_0(x)} + \begin{bmatrix} x^{-2} & 0 \\ 0 & x^{-1} \end{bmatrix} y^{-1} + \begin{bmatrix} x^{-2} & 1 \\ x^{-3} & x^{-1} \end{bmatrix} y^{-2} \ . \tag{3.21}$$

We claim that this matrix cannot be factorized using any one of the univariate PU building blocks in either of the two variables. For this purpose, we consider the following two cases.

♢ *Extraction of a Building Block in* y*:*

To extract a univariate degree-one PU building block from the left side of $\mathbf{P}(x,y)$*, we have to find a unit-norm vector* $\mathbf{v} \in \mathsf{Null}_L(\mathbf{P}_0(x))$*. Any length-two unit-norm vector over* $\mathbb{F}$ *with characteristic two is of the form* $\mathbf{v} = [\, v \quad 1+v \,]^\dagger$ *for some* $v \in \mathbb{F}$*. One can easily verify that such a vector does not exist. For a similar reason, we cannot extract a univariate degree-one PU building block from the right side of* $\mathbf{P}(x,y)$*.*

The next step is attempting to extract a univariate degree-2τ *building block. This is possible only when there exist a nonzero self-orthogonal vector in one of the two null-spaces of* $\mathbf{P}_0(x)$*. Since this is not the case, we cannot extract a univariate degree-*2τ *building block from either side of* $\mathbf{P}(x,y)$*.*

♢ *Extraction of a Building Block in* x*:*

To examine the possibility of extracting a univariate building block in x*, we rearrange* $\mathbf{P}(x,y)$ *as follows.*

$$\begin{aligned} \mathbf{P}(x,y) = & \underbrace{\begin{bmatrix} 0 & 1+y^{-2} \\ 0 & 0 \end{bmatrix}}_{\mathbf{Q}_0(y)} + \begin{bmatrix} 0 & 0 \\ 0 & 1+y^{-1}+y^{-2} \end{bmatrix} x^{-1} \\ & + \begin{bmatrix} 1+y^{-1}+y^{-2} & 0 \\ 0 & 0 \end{bmatrix} x^{-2} + \begin{bmatrix} 0 & 0 \\ 1+y^{-2} & 0 \end{bmatrix} x^{-3} \end{aligned} \tag{3.22}$$

Similar to the previous case, there does not exist a unit-norm vector in any one of the two null spaces of $\mathbf{Q}_0(y)$*. Therefore, we cannot extract a univariate degree-one building*

block in x. In the other hand, none of the two null spaces of $\mathbf{Q}_0(y)$ contains a constant nonzero self-orthogonal vector. Hence, we cannot extract a univariate degree-2τ building block in x.

3.6 Applications

In this section, we first briefly review the application of the two-level factorization method in a complete factorization of multivariate PU matrices over the complex field. We will specifically show that every bivariate PU matrix over the complex field can be factorized into the product of PU IIR building blocks. We will also, very briefly, explain the application of the proposed factorization technique to bivariate IIR PU matrices over the complex field.

As the next application of the bivariate PU building blocks, introduced in Section 3.3, we explain how to design two-dimensional self-dual codes.

3.6.1 Factorization of Bivariate PU Matrices over $\mathbb{C}$

The factorization problem over the complex field $\mathbb{C}$ is much easier than that over finite fields since there are no self-orthogonal nonzero vectors over $\mathbb{C}$. It is shown in [186] that all matrix polynomials in $\mathsf{PU}_M(\mathbb{C}[x^{-1}])$ (FIR PU matrices), for any $M \in \mathbb{N}$, are generated by the degree-one FIR building block

$$\mathbf{B}_{\mathsf{FIR}}(x;\, \mathbf{v}) := \mathbf{I} - \mathbf{v}\,\mathbf{v}^\dagger + \mathbf{v}\,\mathbf{v}^\dagger\, x^{-1} \quad \in\ \mathsf{PU}_M(\mathbb{C}[x^{-1}]) \tag{3.23}$$

in which $\mathbf{v} \in \mathbb{C}^M$ is an arbitrary unit-norm vector. Over the complex field, it makes sense to study IIR PU matrices. It is proved in [186] that all matrices in $\mathsf{PU}_M(\mathbb{C}(x^{-1}))$, for any $M \in \mathbb{N}$, can be generated by the degree-one IIR building block

$$\mathbf{B}_{\mathsf{IIR}}(x;\, \mathbf{v}, a) := \mathbf{I} - \mathbf{v}\,\mathbf{v}^\dagger + \mathbf{v}\,\mathbf{v}^\dagger \left(\frac{-\overline{a} + x^{-1}}{1 - a x^{-1}}\right) \quad \in\ \mathsf{PU}_M(\mathbb{C}(x^{-1}))\ , \tag{3.24}$$

where $\mathbf{v} \in \mathbb{C}^M$ is an arbitrary unit-norm vector and $a \in \mathbb{C}$ is an arbitrary complex number. Using the two-level factorization method, introduced in the previous section, we have proposed a complete factorization technique for all matrices in $\mathsf{PU}_M(\mathbb{C}[x^{-1}, y^{-1}])$ [57, 58].

Precisely, it is shown that an arbitrary bivariate matrix $\mathbf{P}(x,y) \in \mathsf{M}_M(\mathbb{C}[x^{-1},y^{-1}])$ is PU if and only if it can be factorized as

$$\mathbf{P}(x,y) = \left[\frac{1}{\alpha_{N_y}(x)}\,\mathbf{B}_1\Big(y;\,\mathbf{v}_{N_y}(x)\Big)\right]\cdots\left[\frac{1}{\alpha_1(x)}\,\mathbf{B}_1\Big(y;\,\mathbf{v}_1(x)\Big)\right]\left[\frac{1}{\prod_{i=1}^{N_y}\alpha_i(x)}\,\mathbf{F}(x)\right]\ . \quad (3.25)$$

Here, $N_y = \mathsf{Deg}_y\mathbf{P}$ and $\mathbf{B}_1\Big(y;\,\mathbf{v}(x)\Big)$ is the bivariate PU degree-one building block defined as

$$\mathbf{B}_{\mathsf{FIR}}\Big(y;\,\mathbf{v}(x)\Big) := \alpha(x)\mathbf{I} - \mathbf{v}(x)\,\mathbf{v}^{\dagger}(x) + \mathbf{v}(x)\,\mathbf{v}^{\dagger}(x)\,y^{-1} \quad (3.26)$$

in which $\mathbf{v}^{\dagger}(x)\,\mathbf{v}(x) =: \alpha(x) \in \mathbb{C}[x^{-1}] \setminus \{\,0\,\}$. Moreover, in (3.25), $\frac{1}{\prod_{i=1}^{N_y}\alpha_i(x)}\,\mathbf{F}(x) \in \mathsf{PU}_M(\mathbb{C}(x^{-1}))$ is an IIR PU matrix. Using a version of Lemma 3.1.1 extended to IIR matrices, every term in (3.25) (except the last one) can be further factorized. Eventually, the following theorem is proved in [58].

THEOREM 3.6.1. *Every matrix polynomial* $\mathbf{P}(x,y) \in \mathsf{PU}_M(\mathbb{C}[x^{-1},y^{-1}])$ *can be factorized as*

$$\mathbf{P}(x,y) = x^m \prod_{i=1}^{N} \mathbf{A}_i\,\mathbf{B}_{\mathsf{IIR}}(x;\,\mathbf{v}_i,a_i)\,\mathbf{B}_{\mathsf{IIR}}(y;\,\mathbf{u}_i,b_i)\ ,$$

where $m \in \mathbb{Z}$ *and* $N \in \mathbb{N}$. *Moreover, for all* $i \in [\,N\,]$, $\mathbf{A}_i$ *is either the identity matrix or a unitary matrix,* $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{C}^M$ *such that each one of them is either a unit-norm vector or the zero vector, and* $a_i, b_i \in \mathbb{C}$ *are arbitrary complex numbers.*

This level-by-level factorization method is extended to matrices in $\mathsf{PU}_M(\mathbb{C}[x_1^{-1},\ldots,x_n^{-1}])$ for any $n \in \mathbb{N}_{\geq 3}$ [58]. For $n \geq 3$, this technique does not give a complete factorization such as the one in Theorem 3.6.1.

3.6.2 Error-Control Coding

Using the synthesis bank of a bivariate two-channel filter bank over a finite field, a two-dimensional filter-bank code (TDFBC) is designed in [171, 172]. The encoder structure of the TDFBC is depicted in Figure 3.3. In this figure, $\mathbf{M}$ is the upsamlping matrix with $\det\mathbf{M} = 2$ and $\lambda(\mathbf{n})$ is a pre-filter added to find codes with good performances. It is shown in [172] that the TDFBC is lattice cyclic if and only if the dimensions of the codeword arrays

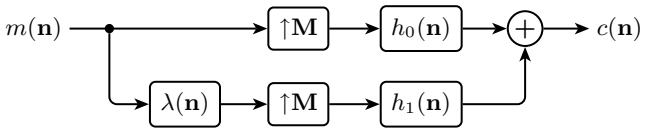


Figure 3.3: Filter bank structure of the half-rate encoder of the TDFBC.

are both even. Because of the lattice-cyclic property of the TDFBC, the encoding and the syndrome computing can be simplified to calculating two-dimensional circular convolutions. The syndrome generator of this code can be realized by the analysis bank of a filter bank as shown in Figure 3.4. The TDFBC is capable of correcting any error burst of pattern $N_1 \times N_2/2$ and $N_1/2 \times N_2$, where it is assumed the filters in the employed filter bank have supports of size $N_1 \times N_2$. The bivariate PU building blocks introduced in Section 3.3 are used to design the bivariate two-channel filter bank employed in the code design.

3.7 Summary

The factorization of bivariate, two-channel, FIR paraunitary filter banks was discussed in this chapter. In our approach, we considered a bivariate FIR PU matrix as a univariate polynomial whose coefficients are matrices with polynomial entries. Using this representation, we were able to extend the univariate factorization methods to the bivariate case. We also generalized the definition of PU matrices to the ring of polynomials. Using this new definition, we presented two fully-parameterized bivariate building blocks that are PU

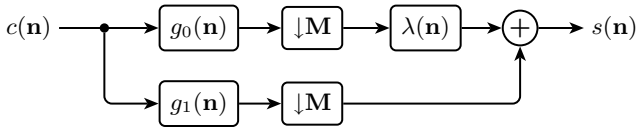


Figure 3.4: Syndrome generator of the TDFBC.

in the ring of polynomials. We developed a method for the first-level factorization of all bivariate PU matrices. This factorization represents any bivariate PU matrices in terms of elementary PU building blocks with respect to one of the variables. The resulting factors are not necessarily FIR with respect to both variables. Hence, further factorization with respect to another variable is not always possible. However, under some conditions, we provided a second-level factorization for these factors. We also applied the factorization to the bivariate PU matrices over the complex field a step forward in a long-standing open research problem. In addition, using the introduced bivariate building blocks, we designed two-dimensional self-dual codes using the synthesis bank of a bivariate filter bank over the finite field.

Part II

Multivariate Cryptography

CHAPTER 4

Introduction

4.1 Historical Background and Motivation

The application of cryptography to ensure privacy depends on the assumption that the communicating parties share a key that is known to no one else. The secret key is traditionally sent in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in the communication era in which people from across the globe communicate using the modern technologies. However, it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by the old-fashion key-distribution methods is a major barrier to the transfer of business communications specially after the advent of e-commerce. Public-key cryptography, proposed by Diffie and Hellman [67] in 1976, provides a practical solution to this problem [137, 16]. A *public-key cryptosystem* is a triple $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of probabilistic polynomial-time algorithms satisfying the following conditions [95].

Key Generation Algorithm: This is a probabilistic expected polynomial-time algorithm KeyGen that on an input $\mathbf{k} \in \mathfrak{K}$ (the security parameter), where $\mathfrak{K}$ is a finite key space, produces the public key $\mathcal{E}$ and the secret key $\mathcal{D}$. The size of the key space $\mathfrak{K}$ must be large enough to make the exhaustive key search infeasible.

Encryption Algorithm: This is a probabilistic polynomial-time algorithm Enc that takes as inputs the security parameter $\mathbf{k}$, the public key $\mathcal{E}$, and a fixed-length plaintext $\mathbf{x} \in \mathfrak{X}$ to generate as output a fixed-length ciphertext $\mathbf{y} = \mathsf{Enc}(\mathbf{k}, \mathcal{E}, \mathbf{x}) \in \mathfrak{Y}$. Here, $\mathfrak{X}$

and $\mathfrak{Y}$ are the plaintext and the ciphertext spaces, respectively.

Decryption Algorithm: This is a probabilistic polynomial-time algorithm Dec that takes as inputs the security parameter $\mathbf{k}$, the secret key $\mathcal{D}$, and a ciphertext $\mathbf{y} \in \mathfrak{Y}$ to generate a plaintext $\mathbf{x}' = \mathsf{Dec}(\mathbf{k}, \mathcal{D}, \mathbf{y}) \in \mathfrak{X}$ such that for all $\mathbf{x} \in \mathfrak{X}$ and for all $\mathbf{y} \in \mathfrak{Y}$, the probability $\mathsf{Prob}(\mathsf{Dec}(\mathbf{k}, \mathcal{D}, \mathbf{y}) \neq \mathbf{x})$ is negligible.

We note that the public key $\mathcal{E}$ is made available to everybody including an adversary. Nevertheless, the secret key $\mathcal{D}$ is known only to legitimate parties. The key generation algorithm should be designed such that the public key does not leak any information about the secret key with high probability.

A public-key cryptosystem significantly simplifies the key distribution problem. Each user u generates a pair of inverse transformations $\mathsf{Enc}_{\mathbf{k}_u}$ and $\mathsf{Dec}_{\mathbf{k}_u}$ by selecting a secret key $\mathbf{k}_u$, keeps $\mathbf{k}_u$ and $\mathsf{Dec}_{\mathbf{k}_u}$ secret, and publishes the encryption algorithm $\mathsf{Enc}_{\mathbf{k}_u}$ in a public directory along with his identification information. Any network user can send a secret message to u over an insecure channel using the public encryption algorithm. However, u is the only one who can decrypt the secret message. This strategy can be used for key distribution. A user A willing to communicate with user B, randomly picks a secret key $\mathbf{k}_{AB}$ and sends it to B using B's public encryption algorithm. Upon receiving $\mathsf{Enc}_{\mathbf{k}_B}(\mathbf{k}_{AB})$, user B obtains $\mathbf{k}_{AB}$ using his knowledge of the secret key $\mathbf{k}_B$. An eavesdropper has only access to $\mathsf{Enc}_{\mathbf{k}_B}(\mathbf{k}_{AB})$ from which it is infeasible to obtain $\mathbf{k}_{AB}$ without knowledge of $\mathbf{k}_B$.

The public-key cryptosystem proposed by Diffie and Hellman in [67] is based on the following problem.

PROBLEM 4.1.1 (Discrete Logarithm). *Let $(\mathcal{G}, \cdot)$ be a cyclic group with a generator g and order $N = |\mathcal{G}|$. Every element $x \in \mathcal{G}$ can be represented as $x = g^a$ for a unique $a \in \mathbb{Z}_N$. Define a function $\mathsf{DL} : \mathcal{G} \rightarrow \mathbb{Z}_N$ that maps any $x \in \mathcal{G}$ to a unique integer $a \in \mathbb{Z}_N$ such that $x = g^a$. For an arbitrary $x \in \mathcal{G}$, calculate $\mathsf{DL}(x)$.*

The function DL in this problem is called the discrete logarithm function and the integer a is referred to as the discrete logarithm of x. The complexity of the best currently

known algorithm to provide an answer to Problem 4.1.1 is exponential in N, the size of the underlying group. Thus, for large groups, this problem cannot be feasibly answered, and it is considered as *computationally hard*.

To exchange keys using the Diffie-Hellman system, two users A and B willing to communicate choose secret integers a and b, compute g^a and g^b, respectively, and exchange them over an insecure channel. The user A upon receiving g^b raises that to power a to obtain g^{ba}. Similarly, user B calculates g^{ab}. Since $g^{ab} = g^{ba}$, the two users have come up with the same information that they use as a secret key. An eavesdropper only knows g^a and g^b. Nevertheless, since the discrete logarithm problem is computationally difficult, it is infeasible for her to obtain a or b.

At the core, public-key cryptography is based on the mathematical notion of *trapdoor one-way function* (TOWF). A function $\Psi : \mathcal{D} \rightarrow \mathcal{R}$ is a TOWF if for every $x \in \mathcal{D}$, the computation of $\Psi(x)$ is computationally efficient. However, for almost every $y \in \mathcal{R}$, the inversion $f^{-1}(y)$ is efficiently computable if and only if information of a trapdoor is available. Otherwise, it is infeasible to solve the equation $y = f(x)$ for x given y. A TOWF can be used to produce a public-key cryptosystem [67].

Another application of TOWFs is in entity authentication. Assume a network user B receives a message m from another user A. To provide some cryptographic evidence that m was indeed generated by A, the user A employs a TOWF Ψ. Using her knowledge of the trapdoor, A solves the equation $m = \Psi(s)$ for s and sends m and s along with Ψ to B. Upon receiving this information, B verifies the authenticity of the message by verifying $m = \Psi(s)$. Since without knowledge of the trapdoor information, calculation of s is infeasible, no other user in the network could have generated an s that satisfies $m = \Psi(s)$. Hence, this is a legitimate procedure for entity authentication.

A one-way function (OWF) without a trapdoor has also cryptographic applications in entity authentication. For example, consider the computer login scenario. When a user first enters his password PW, the computer stores $\Psi(\mathsf{PW})$ instead of PW assuming Ψ is an OWF. At each successive login, the computer calculates $\Psi(\mathsf{PW}')$, where PW' is the

proffered password, and compares it with $\Psi(\mathsf{PW})$. If and only if these two are the same, the user is accepted as being authentic.

4.2 RSA

The first practical implementation of a TOWF is RSA that was proposed by Rivest, Shamir, and Adleman [167]. The TOWF employed in RSA is the univariate monomial x^e, for some fixed integer $e \in \mathbb{N}$, over a very large ring such as $\mathbb{Z}_n$. Here, $n = pq$, called the modulus, is a composite integer with large prime factors p and q. The public exponent e is invertible modulo $\phi(n) = (p-1)(q-1)$, where ϕ is the Euler function [169]. The secret trapdoor is the positive integer $d = e^{-1} \mod \phi(n) \in \mathbb{N}$. Precisely, the decryption function is the monomial x^d over the ring $\mathbb{Z}_n$.

The difficulty of inverting this TOWF is related to the difficulty of solving the following problem as will be explained shortly [137, 95].

Problem 4.2.1 (Integer Factorization). *Given an arbitrary composite integer $n \in \mathbb{N}$, factor it into a product of primes.*

The existence of any polynomial-time algorithm for the factorization problem implies that RSA can be feasibly broken. Indeed, an adversary having access to such algorithm can efficiently factor the modulus n into its prime factors p and q. Knowing these factors, the adversary can simply calculate $\phi(n)$ and consequently the decryption exponent d. Nevertheless, it is unclear that breaking RSA is bound to the existence of an efficient factorization algorithm.

In recent years, the limits of the best integer factorization algorithms have been extended greatly due in part to Moore's law and in part to algorithmic improvements [28]. The best integer-factorization algorithm has sub-exponential time-complexity [47]. Hence, the current minimum recommended size for the RSA modulus is 1024 bits. Moreover, as suggested in [173], the minimum size must be 4096 bits by 2015 and 8192 bits by 2025. Two problems in practical implementations of RSA are the key-setup time and the size of

the signature that are too long for resource-limited devices using low-power processors. For example, it takes tens of minutes on a Palm V that uses a 16.6 MHz Dragonball processor to generate 1024-bit RSA key [142]. These problems become more sensible when the modulus size is increased as suggested in [173]. Moreover, there exist attacks on RSA for small values of the secret exponent d [23] or when a small fraction of the private key is known [24].

4.3 Elliptic Curve Cryptography

An instance of the public-key cryptosystem proposed in [67] is obtained when $\mathcal{G}$ is the multiplicative group $\mathbb{Z}_p^\times$ of order $p-1$ for any prime integer p. The advantage of using this group is that the exponentiation can be efficiently performed using the binary representation of the exponent. An algorithm is also proposed by Adleman et al. [2] for computing the discrete logarithm over all finite fields that is conjectured to have sub-exponential time-complexity. Therefore, the idea of using a cyclic subgroup of the group of points on elliptic curves over $\mathbb{F}_p$, for some prime p, was independently proposed by Miller [141] and Koblitz [117]. An elliptic curve is a plane curve defined by an equation of the form $y^2 = x^3 + ax + b$ for fixed $a, b \in \mathbb{F}_p$. The points on this curve form a group with an addition law explained in [118]. The time complexity of all algorithms for solving the discrete logarithm problem over such groups is exponential (except the special case of super singular curves). Thus, comparing to RSA, systems based on the discrete logarithm over elliptic curves are able to maintain the same security level with shorter keys. However, the shortest signature that can be generated using an elliptic curve digital signature algorithm (ECDSA) is 320 bits [8] that is still long for some applications[1]. Moreover, the complexity of the elliptic-curve signature-verification algorithm is high. A comparison between ECDSA and RSA in a field with prime characteristic shows that for practical sizes of fields and moduli, signature verification with ECDSA is 40 times slower than that with RSA [200].

[1]Using Weil paring, a signature scheme capable of producing signatures as short as 160 bits has been proposed in [25].

4.4 Multivariate Cryptography

Considering the shortcomings of the RSA and ECDSA, it would be desirable to have practical cryptosystems based on problems other than the assumptions currently in use. We might be in a safer state against possibilities such as the emergence of an efficient algorithm for factoring or computing discrete logarithms. We note that both the RSA and elliptic curve cryptography (ECC) based cryptosystems employ a monomial of large degree over a very large ring or group. An alternative approach is using polynomials of small degrees over small finite fields. Cryptosystems utilizing this approach fall in the category of *multivariate cryptography* that is considered to be the cryptography of the 21st century [4]. The TOWF in such cryptosystems usually consists of a collection of multivariate polynomials. There are very efficient algorithms for evaluating polynomials at a point. However, the inverse problem, i.e., given the values of the polynomials, solve the system of multivariate polynomial equations for a point that satisfies all equations, is believed to be difficult [31, 121, 1]. Mathematically, this problem is stated as follows.

PROBLEM 4.4.1 (Solving Systems of Polynomial Equations). *Fix two integers $n, m \in \mathbb{N}$ and a finite field $\mathbb{F}$. Uniformly at random make the following choices.*

(i) m integers $N_1, \ldots, N_m \in \mathbb{N}$ and m finite-field elements $y_1, \ldots, y_m \in \mathbb{F}$

(ii) for every $i \in [\,m\,]$, N_i finite-field elements $\alpha_{i,1}, \ldots, \alpha_{i,N_i} \in \mathbb{F}$ and N_i integer vectors $\mathbf{a}_{i,1}, \ldots, \mathbf{a}_{i,N_i} \in (\mathbb{Z}_{\geq 0})^n$ each of length n

Construct m multivariate polynomials $f_1, \ldots, f_m$ such that[2] $f_i(\mathbf{x}) = \sum_{j=1}^{N_i} \alpha_{i,j}\, \mathbf{x}^{\mathbf{a}_{i,j}}$ *for all $i \in [\,m\,]$. Solve the system of equations*

$$\begin{cases} f_1(x_1, \ldots, x_n) = y_1 \\ \qquad\qquad \vdots \\ f_m(x_1, \ldots, x_n) = y_m \end{cases} \tag{4.1}$$

for $x_1, \ldots, x_n$.

[2]The notation $\mathbf{x}^{\mathbf{a}}$ is formally defined in Section 1.1.4.

The classical solution to this problem is computing the Gröbner basis of the ideal ($f_1, \ldots,$ f_m) using a lexicographic order on the terms of multivariate polynomials [14, 45]. Polynomials in the Gröbner basis of a system of polynomial equations have a triangular structure such as the equations obtained after the Gaussian elimination of a system of linear equations. The first algorithm proposed for computing the Gröbner basis is the Büchberger algorithm, which has exponential time-complexity [29, 31]. However, some new algorithms have been proposed recently that are more efficient than the Büchberger algorithm [78, 79].

It can be shown [45] that Problem 4.4.1 is equivalent to the ideal membership problem stated as "Given an ideal $I = (f_1, \ldots, f_m) \subset \mathbb{F}[x_1, \ldots, x_n]$ and an arbitrary polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, determine whether $f \in I$." Therefore, many authors have studied the complexity of the latter [31, 121]. It is shown in [1] that over fields of characteristic two, this problem is NP-hard in the worst case.

FACT 4.4.1 (Computational Complexity of Problem 4.4.1). *The computation of a Gröbner basis over a field of characteristic two is characterized by a space complexity that grows doubly exponentially with the number of variables [1].*

In cryptography, we are interested in the average complexity of problems. It is shown in [13] that computing Gröbner bases in the average has lower complexity single exponential. Precisely, it is shown that in zero-dimensional case[3] and also when the homogenized system has finitely many solutions, the average complexity is[4] $k^{O(1)}\theta^{O(n)}$, where k is the number of equations, θ is the maximum degree of equations, and n is the number of variables. We note that given a multivariate polynomial $f(x_1, \ldots, x_n)$ with maximum degree θ, it can always be homogenized by rewriting it in $n+1$ variables as $X_0^\theta f(\frac{X_1}{X_0}, \ldots, \frac{X_n}{X_0})$.

The efficiency of multivariate cryptosystems comes from the fact that the underlying algebraic structure (group, field, or ring) has an order much smaller than the order of the ring used in RSA or the order of cyclic subgroups of points on elliptic curves. As a result, such cryptosystems can be implemented in resource-limited environments such as smart

[3] When the system of polynomial equations has finitely many solutions.
[4] This case gives the lowest possible complexity.

cards, cell phones, and PDAs. An implementation of an instance of the HFE cryptosystem over $\mathsf{GF}(64)$ on a Pentium-III 730MHz running Java 1.3 is proposed in [201]. The reported running times for the key generation, encryption, and decryption are 210μs, 14μs, and 56μs, respectively. These numbers are much smaller than those of RSA or ECC based systems. As another example, a fast implementation of the signature scheme SFLASH on a low-cost eight-bit 10MHz smart card (without coprocessor) is proposed in [3]. The generation of a 259-bit signature using SFLASH takes only 59ms. However, generating signatures of lengths 1024 and 382 bits on a similar 10MHz card with coprocessor using RSA and ECDSA takes 111ms and 180ms, respectively.

Considering the attractive features of multivariate cryptography, it is desirable to investigate the possibility of implementing such techniques in designing symmetric cryptosystems. To address this issue, in Chapter 5, we propose a multivariate stream cipher based on wavelet transform. Using multivariate PU matrices, we propose general frameworks for designing multivariate asymmetric cryptosystems and signature schemes in Chapter 6.

CHAPTER 5

Wavelet Self-Synchronizing Stream-Cipher

Stream ciphers form an important class of encryption algorithms. They encrypt individual characters (or digits in digitized data) of a plaintext message one at a time using an encryption transformation which varies with time [170, 179, 137, 68]. By contrast, block ciphers simultaneously encrypt blocks of characters (digits) of a plaintext message using a fixed encryption transformation. Stream ciphers are generally faster than block ciphers in hardware and have less complex hardware circuitry. In addition, they are more appropriate (e.g., in some telecommunications applications) when buffering is limited or when characters must be individually processed as they are received. Because they have limited or no error propagation, stream ciphers may also be advantageous in situations where transmission errors are highly probable.

As we will explain later, most of the previous work in the design of stream ciphers is based on linear feedback shift registers (LFSRs). The secret key determines the initial states of the LFSRs employed in the design. To mask the linear recurrence relations at the LFSR outputs, nonlinear Boolean functions are used. Many powerful attacks have been developed for systems designed based on this approach. Therefore, it is desirable to start afresh and propose a new methodology for the stream cipher design. Considering the attractive features of multivariate cryptosystems, we investigate the possibility of developing a multivariate self-synchronizing stream cipher in this chapter. Using the wavelet transform over finite fields as a sequence transformer, we propose a novel technique for the design of self-synchronizing stream ciphers in this chapter.

5.1 Background Review

Vernam cipher is the first stream cipher in which the plaintext stream $p \in \mathbb{F}_2^*$, the key stream $k \in \mathbb{F}_2^*$, and the ciphertext stream $c \in \mathbb{F}_2^*$ are related as $c_n = p_n + k_n$ for all $n \in \mathbb{Z}_{\geq 0}$. The decryption is performed as $p_n = c_n + k_n$. If the key stream is employed only once and its digits are generated randomly and independently, then the Vernam cipher is called *one-time pad*.

Shannon proved that a necessary condition for a symmetric-key encryption scheme to be "perfectly secure" is that $H(K) \geq H(P)$, where $H(\cdot)$ is the entropy function and P and K are random variables denoting the plaintext and the secret key, respectively [176]. In other words, the uncertainty of the secret key must be at least as great as the uncertainty of the plaintext. With this result, the one-time pad is perfectly secure, i.e., the scheme is unconditionally secure regardless of the computational power of the adversary and the distribution of the plaintext.

The major drawback of the one-time pad is that the key stream must be at least as long as the plaintext stream. This observation motivates the design of stream ciphers in which the key stream is pseudorandomly generated from a short secret key used as a seed to a key generator. Although this approach does not offer perfect secrecy since $H(K) \ll H(P)$, it is intended to capture the spirit of one-time pad by designing a key generator that outputs a sequence which appears random to a computationally-bounded adversary.

5.1.1 Classification of Stream Ciphers

Stream ciphers are commonly classified as *synchronous* and *self-synchronizing* (or *asynchronous*) [170, 137, 168, 164]. The main property of the synchronous stream ciphers is that both the sender and the receiver must be synchronized to allow proper operation. If the synchronization is lost due to the insertion or deletion of the ciphertext digits by an adversary or the transmission channel, the decryption fails. Hence, additional techniques for the re-synchronization are required. Other two important properties of synchronous

stream ciphers are:

No error propagation: The modification of a ciphertext digit during the transmission does not affect the decryption of other ciphertext digits.

Active attacks: The absence of error propagation has a positive and a negative side in the presence of an active attacker who may insert, delete, or replay ciphertext digits. A careless attacker may cause immediate loss of synchronization that will be detected at the decryption time. However, an intelligent adversary can selectively modify ciphertext digits and forge a message without interfering with the synchronization. Therefore, additional mechanisms must be employed to provide data origin authentication and data integrity guarantee.

As their name implies, self-synchronization is possible in SSCs if ciphertext digits are deleted or inserted. This is because the decryption function depends only on a limited number of previous ciphertext digits. Such ciphers are capable of automatically re-establishing proper decryption after the loss of the synchronization. This property makes SSCs suitable for applications such as pay TV in which a user joining the network at any time must be able to decrypt the stream of enciphered data. Other properties of SSCs are as follows [137]:

Limited error propagation: If a ciphertext digit is modified during the transmission, the decryption of up to a fixed number of ciphertext digits might be incorrect after which the correct decryption resumes. This property has its own pros and cons. The advantage is robustness to channel noise and other environmental effects specially in wireless transmission. The disadvantage, in comparison with synchronous stream ciphers, is limited chance for detecting adversarial activities that put the system out of synchronization.

Active attacks: The presence of error propagation implies that any modification of ciphertext digits (without tampering with the system synchronization) by an active

adversary propagates to several other ciphertext digits, thereby improving the likelihood of being detected at the decryption time.

Diffusion of plaintext statistics: Since each plaintext digit influences the entire following ciphertext, the statistical properties of the plaintext are dispersed through the ciphertext. Hence, SSCs may be more resistant to attacks based on plaintext redundancy.

As suggested by Shannon in [176], two main strategies used in the design of cryptographic schemes to prevent statistical analysis are *diffusion* and *confusion*. In diffusion, the redundancy in the message, which is revealed to an adversary through an statistical analysis, is dissipated into long-range statistics. As a result, the adversary has to observe a large amount of the message to effectively mount an attack based on statistical analysis. Confusion is intended to make the relationship between the key and the ciphertext as complex as possible. Stream ciphers follow these guidelines as well. In the following, we briefly review the relevant work in the design of stream ciphers.

A Synchronous Stream Ciphers

A synchronous stream cipher is one in which the key stream is generated independent of the plaintext and the ciphertext. Such system can be modeled as a finite-state machine (FSM) [103] consisting of the state space $\mathcal{S}$. It takes a short secret key $\mathbf{k} \in \mathbb{F}^L$, where $\mathbb{F}$ is some finite field of characteristic two and $L \in \mathbb{N}$. The key generator $\mathsf{KeyGen}\colon \mathcal{S} \times \mathbb{F}^L \to \mathbb{F}$ takes $\mathbf{k}$ as a seed and a current state $\varrho_n \in \mathcal{S}$ and outputs the current key stream digit $z_n \in \mathbb{F}$. The key generator is intended to be a pseudo-random number generator with good statistical properties. The initial state $\varrho_0 \in \mathcal{S}$ is determined from the seed $\mathbf{k}$. The current state is updated using a state evolution function $\mathsf{StateEv}\colon \mathcal{S} \times \mathbb{F}^L \to \mathcal{S}$. The encryption function is the XOR of the plaintext stream p with the key stream z digit-by-digit. Mathematically,

this procedure is formulated as

$$\varrho_{n+1} = \mathsf{StateEv}(\varrho_n, \mathbf{k}) \tag{5.1a}$$

$$z_n = \mathsf{KeyGen}(\varrho_n, \mathbf{k}) \tag{5.1b}$$

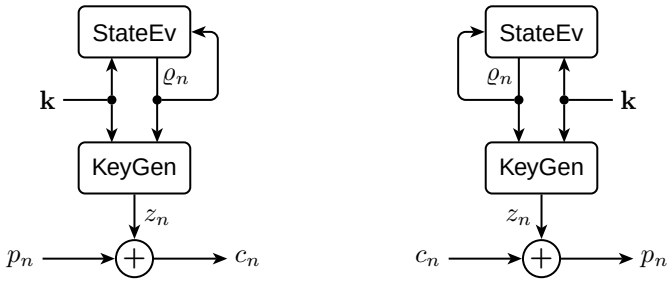
$$c_n = p_n + z_n \tag{5.1c}$$

for all $n \in \mathbb{Z}_{\geq 0}$. The encryption and decryption processes are depicted in Figure 5.1.

In the traditional approach to the deign of synchronous stream-ciphers, LFSRs are used in the key generator since they are well-suited for hardware implementation, produce sequences with large periods and good statistical properties, and can be analyzed using algebraic tools [170, 137, 164]. As shown in Figure 5.2, an LFSR of length L consists of L memory units $D_0, \ldots, D_{L-1}$ each capable of storing one bit and having one input and one output. Every LFSR is equipped with a clock that controls the flow of the data in the register. During each unit of time (defined by a clock signal), D_i takes the content of D_{i+1} for all $i \in (\, L-1\,)$. Let s_{n-i} be the content of the memory unit D_{L-i} at an arbitrary time instance $n \in \mathbb{Z}_{\geq 0}$ for all $i \in [\, L\,]$. The memory unit D_{L-1} takes

$$s_n = \sum_{i=1}^{L} a_{L-i}\, s_{n-i} \ , \tag{5.2}$$

where $a_i \in \mathbb{F}_2$ for all $i \in (\, L\,)$. The vector $\varrho_n := (\, s_{n-1}, \ldots, s_{n-L}\,) \in \mathbb{F}_2^L$ is the *state* of the LFSR at the time instance $n \in \mathbb{Z}_{\geq 0}$. Given the *initial state* ϱ_0 of an LFSR, its output is uniquely determined.



(a) Encryption. (b) Decryption.

Figure 5.1: General model of synchronous stream ciphers.

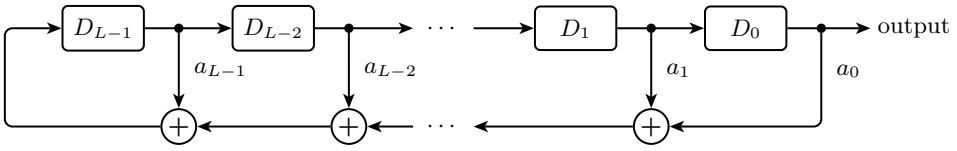


Figure 5.2: Linear feedback shift register of length L.

Equation (5.2) is called an *L-th order linear recurring relation*, and the corresponding sequence $s = (s_0, s_1, \dots)$ is called an *L-th order linear recurring sequence*. The monic polynomial

$$c(x) = x^L - a_{L-1}x^{L-2} - a_{L-2}x^{L-2} - \cdots - a_1 x - a_0 \quad \in \mathbb{F}_2[x] \tag{5.3}$$

is called the *characteristic polynomial* of the linear recurring sequence. For every linear recurring sequence, there exists a unique characteristic polynomial with minimum degree, which is called the *minimal polynomial* of the sequence [125]. It can be shown that every linear recurring sequence is periodic [125]. Since the goal is generating a pseudo-random sequence, the period should be maximum. It can be proved that if the minimal polynomial in (5.3) is a primitive polynomial, then the sequence has the maximum possible period $2^L - 1$ [125, 137]. Moreover, each of the $2^L - 1$ nonzero initial states generate a sequence with maximum order.

An LFSR should never be used by itself as a key generator since its output sequence is easily predictable using a known or chosen plaintext attack [125]. One technique for destroying the inherent linearity in LFSRs is to include nonlinear Boolean functions in the design of the key generator. Based on the structural combination of LFSRs and nonlinear functions, there are two classical models for memoryless synchronous stream ciphers: the nonlinear filter (NF) model and the nonlinear combiner (NC) model [68, 170, 137]. In the former, one LFSR of length L is used. Let $\varrho_n = (s_{n-1}, \dots, s_{n-L}) \in \mathbb{F}_2^L$ be the current state of the LFSR at time instance $n \in \mathbb{Z}_{\geq 0}$. The key stream digit z_n at any time instance n is the output of a nonlinear Boolean function $f\colon \mathbb{F}_2^L \to \mathbb{F}_2$ fed with ϱ_n. The initial state ϱ_0 is the secret key $\mathbf{k} \in \mathbb{F}_2^L$ of the system. Hence, the size of the key space is 2^L.

In the NC model, several LFSRs are used. Let $(s^i_{n-1}, \ldots, s^i_{n-L_i}) \in \mathbb{F}_2^{L_i}$ be the current state of the i-th LFSR for all $i \in [N]$, where $N \in \mathbb{N}$ is the number of LFSRs and L_i is the length of the i-th LFSR. The key stream digit z_n at any time instance n is the output of a nonlinear Boolean function $f\colon \mathbb{F}_2^N \to \mathbb{F}_2$ for the input $(s^1_{n-L_1}, \ldots, s^N_{n-L_N})$. The secret key determines the initial states of all LFSRs. The length of the secret key and the size of the key space are $L = \sum_{i=1}^N L_i$ and 2^L, respectively.

In an LFSR-based stream cipher, the coefficients of the minimal polynomial, i.e., the constants $a_0, \ldots, a_{L-1}$ in (5.3), may depend on the secret key as well as the initial state of the LFSR. Although this strategy adds to the security, it may not be cost effective. The first reason is that the minimal polynomial is usually primitive to achieve the maximum period. Therefore, the number of possibilities is limited. The second reason is that the minimal polynomial of every L-th order linear recurring sequence can be determined using the Berlekamp-Massey algorithm from only $2L$ consecutive elements of that sequence [125].

Another approach to destroying the linearity of LFSRs is to use clock-controlled shift registers [97]. In these systems, some of the registers are stepped or clocked in an irregular manner under the control of other registers, so that attacks based on the regular motion of the registers can be foiled. There are many variants and designs for clock-controlled generators. In the following, we briefly review some of them and also some alternative designs for pseudo-random generators. A good review of different designs can be found in [168]. In all the following descriptions, we assume all sequences belong to $\mathbb{F}_2^*$, and they are indexed by nonnegative integers $\mathbb{Z}_{\geq 0}$ unless otherwise stated.

Stop-and-Go Generator: Proposed in [17], this generator consists of two LFSRs L_A and L_B that generate sequences a and b, respectively. Let $k_n := \sum_{i=0}^{n-1} b_i \in \mathbb{Z}_{\geq 0}$, where the summation is carried out over the ring of integers. Then, $z_n = a_{k_n}$ for all $n \in \mathbb{Z}_{\geq 0}$, where z is the output sequence.

Alternating Step Generator: This generator, proposed in [101], is very similar to the stop-and-go generator. It consists of three LFSRs C, L, and $\overline{\mathsf{L}}$. In the proposed structure,

C clocks L and $\overline{\mathsf{L}}$ when it outputs 1 and 0, respectively. The output sequence is the XOR of the output sequences of L and $\overline{\mathsf{L}}$. Let c, x, and $\overline{x}$ be the sequences generated by C, L, and $\overline{\mathsf{L}}$, respectively, when they are independently clocked. In addition, let $k_n := \sum_{i=0}^{n-1} c_i \in \mathbb{Z}_{\geq 0}$ and $\overline{k}_n := n - k_n \in \mathbb{Z}_{\geq 0}$ for all $n \in \mathbb{Z}_{\geq 0}$, where all calculations are carried out over the ring of integers. Then, the key stream digit z_n at an arbitrary time instance n is

$$z_n = x_{k_n} + \overline{x}_{\overline{k}_n} \qquad \forall n \in \mathbb{Z}_{\geq 0} \ . \tag{5.4}$$

Shrinking Generator: The shrinking generator employs two sources of pseudo-random bits (two LFSRs) to generate a third one with better statistical properties [37]. The output sequence is a subsequence of the first source such that its elements are chosen according to the positions of 1's in the second source. In other words, the output sequence is a shrunken version of one of the sequences. Let a and b be sequences generated by the first and the second sources, respectively, and z be the output sequence. At any time instance $n \in \mathbb{Z}_{\geq 0}$, the output bit is $z_n = a_{k_n}$, where k_n is the position of the n-th 1 in the sequence b.

Self-Shrinking Generator: This is a variant of the shrinking generator that employs only a single LFSR [135]. Let $a = (a_0, a_1, a_2, \dots) \in \mathbb{F}_2^*$ be the sequence generated by the LFSR when it is clocked normally. The self-shrinking operation considers this sequence as one of pairs of bits, i.e., $b = (\,(a_0, a_1), (a_2, a_3), \dots) \in (\mathbb{F}_2 \times \mathbb{F}_2)^*$. If a pair $(a_{2n}, a_{2n+1}) \in \{\,(1,0), (1,1)\,\}$, then a_{2n+1} is the value of the output sequence at the current time instance. Otherwise, if $(a_{2n}, a_{2n+1}) \in \{\,(0,0), (0,1)\,\}$, the pair (a_{2n}, a_{2n+1}) is discarded. To put this idea into mathematical language, let k_n be position of the n-th 1 in the subsequence $(a_0, a_2, a_4, \dots)$. Then, $z_n = a_{2k_n+1}$, where z is the output sequence.

Generalized Self-Shrinking Generator: As its name implies, this is a generalization of the self-shrinking generator [104]. Let a be a sequence generated by an LFSR when clocked normally. Suppose $g_0, \dots, g_{L-1} \in \mathbb{F}_2$ are fixed coefficients. Generate the sequence b as $b_n = \sum_{i=0}^{L-1} g_i\, a_{n-i}$, where the assumption is $a_n = 0$ for all $n \in \mathbb{Z}_{<0}$. The shrunken version of b is the sequence generated by the generalized delf-shrinking generator. It is

shown in [104] that this sequence is balanced and has good statistical properties.

Generators Based on Cellular Automata: A pseudo-random number generator based on cellular automata was proposed in [202]. A cellular automata is an extension of simple shift registers in which every memory cell has access to the contents of its two neighbor cells. Consider an array of L memory cells holding the values $a_n^{(0)}, a_n^{(1)}, \ldots, a_n^{(L-1)} \in \mathbb{F}_2$ at an arbitrary time instance $n \in \mathbb{Z}_{\geq 0}$. In the generator proposed in [202], the content of each memory cell is updated as

$$a_{n+1}^{(i)} = a_n^{(i-1)} + a_n^{(i)} + a_n^{(i+1)} + a_n^{(i)}\, a_n^{(i+1)} \qquad \forall i \in (\, L\,) \quad , \quad \forall n \in \mathbb{Z}_{\geq 0} \ , \tag{5.5}$$

where we assume $a_n^{(-1)} = a_n^{(L-1)}$ and $a_n^{(L)} = a_n^{(0)}$. The secret key determines the initial state of the cellular automata.

LILI Key-Stream Generator: This generator is constructed by two nonlinear-filter generators such that the output of one irregularly clocks the other [178]. The clock-control generator consists of an LFSR of length α and a nonlinear Boolean function $f_c \colon \mathbb{F}_2^\alpha \to \mathbb{F}_2$. Similarly, the data generator consists of an LFSR of length β and a nonlinear Boolean function $f_c \colon \mathbb{F}_2^\beta \to \mathbb{F}_2$. At an arbitrary time instance $n \in \mathbb{Z}_{\geq 0}$, let $\varrho_n^c \in \mathbb{F}_2^\alpha$ and $\varrho_n^d \in \mathbb{F}_2^\beta$ be the state vectors of the clock control and data LFSRs, respectively. When normally clocked, the two generators output $c_n = f_c(\varrho_n^c)$ and $d_n = f_d(\varrho_n^d)$. The output of LILI is the sequence z such that $z_n = d_{k_n}$, where $k_n := \sum_{i=0}^{n-1} c_i$.

Generators Based on OFB Mode of Block Ciphers: Block ciphers in the output feedback (OFB) mode can be used as pseudo-random number generators [137, 81]. This mode of operation is demonstrated in Figure 5.3. In this figure, $\mathbf{k}$ is the secret key of the block cipher and IS is the initial state of the memory.

RC4: This stream cipher, developed by Ron Rivest in 1987, is a variable-key-size cipher suitable for fast bulk encryption [168]. RC4 is very compact in terms of code size, and it is particularly suitable for byte-oriented processors. It can encrypt at speeds of around

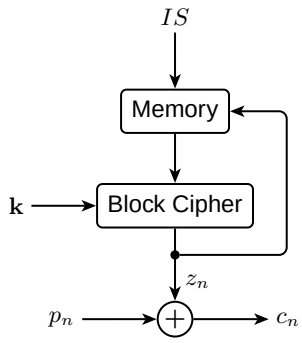


Figure 5.3: OFB mode of block ciphers.

1 Mbyte/sec on a 33 MHz machine. RC4 has a secret internal state that is a permutation $\sigma \in \mathfrak{S}_{2^n}$ consisting of 2^n different n-bit words for some fixed $n \in \mathbb{N}$ [173, 130]. The initial state is derived from a variable-size secret key $\mathbf{k} \in \mathbb{Z}_{2^n}^L$. In practice, n is typically chosen as 8 that provides a huge state of approximate size 1,700 bits. Hence, it is infeasible to mount a time-memory tradeoff attack. In addition, the state evolves in a complex and nonlinear way that makes it difficult to combine partial information about states that are far away in time. RC4 can be described as in Algorithm 5.1. The total memory required to implement this algorithm is $n2^n + 2n$ bits.

B Self-Synchronizing Stream Ciphers

The second class of stream ciphers is self-synchronizing in which the key stream is generated as a function of the secret key $\mathbf{k} \in \mathbb{F}^L$ and a fixed number of previous ciphertext digits. The encryption function of an asynchronous stream cipher is described by the equations

$$\varrho_n = (c_{n-1}, \ldots, c_{n-L}) \tag{5.6a}$$

$$\varrho_{n+1} = \mathsf{StateEv}(\varrho_n, c_n, \mathbf{k} \tag{5.6b}$$

$$z_n = \mathsf{KeyGen}(\varrho_n, \mathbf{k}) \tag{5.6c}$$

$$c_n = p_n + z_n \tag{5.6d}$$

Algorithm 5.1: RC4 key generation.

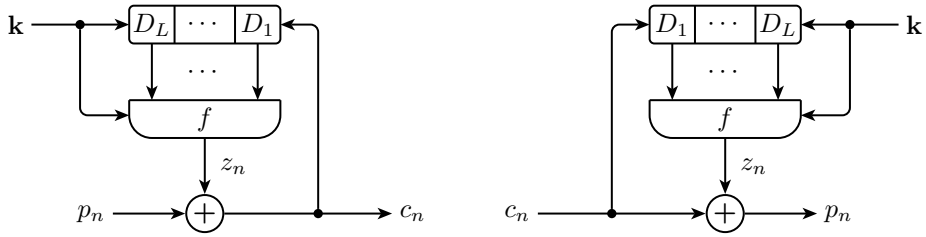
INPUT: Secret key $\mathbf{k} = (k_0, \ldots, k_{L-1}) \in \mathbb{Z}_{2^n}^L$ and the number $N \in \mathbb{N}$ of key stream digits

OUTPUT: Key stream digits $z_0, \ldots, z_{N-1} \in \mathbb{Z}_{2^n}$

1. $\sigma \leftarrow \mathsf{id}$ $\quad \triangleright$ Identity permutation
2. $j \leftarrow 0$
3. **for** $i = 0$ **to** $2^n - 1$ **do**
4. $\quad j \leftarrow \big(j + \sigma(i) + k_{((i))_L}\big) \mod 2^n$
5. $\quad \sigma(i) \leftrightarrow \sigma(j)$
6. **end**
7. $i, j \leftarrow 0$
8. **for** $k = 0$ **to** $N - 1$ **do**
9. $\quad i \leftarrow (i + 1) \mod 2^n$
10. $\quad j \leftarrow (j + \sigma(i)) \mod 2^n$
11. $\quad \sigma(i) \leftrightarrow \sigma(j)$
12. $\quad z_k \leftarrow \sigma\big((\sigma(i) + \sigma(j)) \mod 2^n\big)$
13. **end**

for all $n \in \mathbb{Z}_{\geq 0}$. Here, $L \in \mathbb{N}$ is a fixed integer that determines the number of previous cipher digits on which the next cipher digit depends and $\varrho_n \in \mathbb{F}_2^L$ is the state of the cipher at time instance n. Moreover, $\mathsf{StateEv} : \mathbb{F}_2^L \times \mathbb{F}_2 \times \mathbb{F}_2^L \to \mathbb{F}_2^L$ is the state evolution function and $\mathsf{KeyGen} : \mathbb{F}_2^L \times \mathbb{F}_2^L \to \mathbb{F}_2$ is the key generator. Figure 5.4 shows the canonical representation of SSCs [134]. As the figure shows, the cipher digits are fed back to the system and stored in a shift register of length L. At an arbitrary time instance $n \in \mathbb{Z}_{\geq 0}$, the memory cell D_i stores c_{n-i} for all $i \in [\, L \,]$. The contents of the shift register are fed to a nonlinear Boolean function $f \colon \mathbb{F}_2^L \to \mathbb{F}_2$ that generates the key stream digit z_n. The secret key $\mathbf{k} \in \mathbb{F}_2^L$ determines the initial state of the shift register and also is an input argument to the nonlinear Boolean function f.

Similar to their synchronous counterpart, SSCs are traditionally designed using LFSRs in either the NF or the NC model. It is also possible to design an SSC using any block



(a) Encryption. (b) Decryption.

Figure 5.4: Canonical model of self-synchronizing stream cipher.

cipher in the cipher feedback (CFB) mode as depicted in Figure 5.5.

We note that, as suggested by Shannon, diffusion and confusion are the two main design strategies governing the structures in Figure 5.4:

1. The feedback at the encryption diffuses the statistics of the plaintext at any time instance to all the future ciphertext digits.

2. The invertible transformations map a sequence into another through a set of complex equations. In fact, they are designed to confuse the attacker. In addition, they help to better diffuse the statistical properties of the plaintext into the ciphertext.

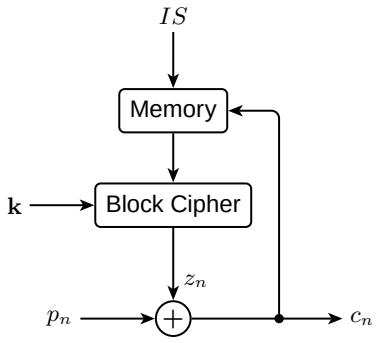


Figure 5.5: CFB mode of block ciphers.

5.2 Wavelet Self-Synchronizing Stream Cipher (WSSC)

Using the DTWT over fields of characteristic two, introduced in Part I, we propose a new SSC [60], which we refer to as wavelet self-synchronizing stream cipher (WSSC). This stream cipher is distinguished from traditional designs by the following differences.

1. While most of the stream ciphers operate on the binary field, the WSSC operates on the Galois field GF(256) to enhance its byte-oriented implementation. Lifting to a higher-order field allows using algebraic techniques that cannot be efficiently used in the binary field.

2. In the design of SSCs, LFSRs are usually employed to realize FSMs. However, in our design, we use DTWTs that are, in fact, sequence transformers. Since wavelets are linear transformations, we also include nonlinear mappings in our design. Our motivations for using the DTWT are as follows.

   a. Wavelets have a rich history in mathematics and signal processing [50, 190, 186, 86]. They are shown to be useful tools for the efficient processing of finite sequences over finite or infinite alphabets.

   b. There are algorithms that are efficient in both software and hardware for the calculation of the wavelet transformation.

   c. The algebraic nature of the method enhances the design and the security analysis of the stream cipher.

3. In contrast to traditional designs in which the secret key determines the initial states of the LFSRs, it is used in our proposed cipher to construct DTWTs. Consequently, most of the traditional attacks, such as correlation and linear attacks, fail in the WSSC.

4. The notion of "round" is usually reserved for block ciphers. A cascade of several basic rounds makes a block cipher resistant to many attacks that intend to propagate some characteristic to the output. In fact, Shannon showed in [176] that when the plaintext and the ciphertext symbols both belong to the same alphabet, the iterative usage of a

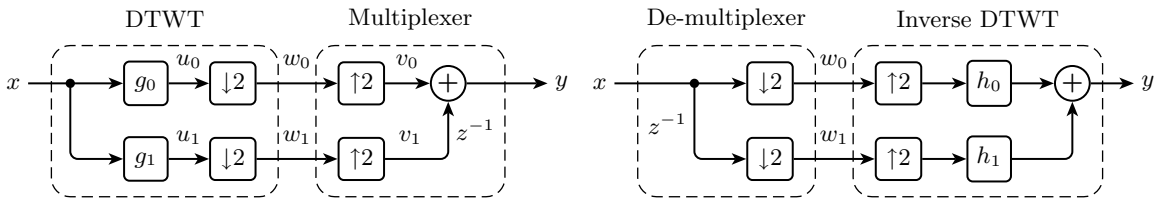
cipher improves the security. We adopt this notion in the design of the WSSC, where one round is structured as in Figure 5.4.

Our goal is using the DTWT and its inverse as sequence transformers. For this purpose, we may use the analysis and synthesis banks of the filter bank structure in Figure 2.1 as the DTWT and its inverse, respectively. To setup the DTWT and its inverse, one employs Algorithm 2.1 and Algorithm 2.2 in Chapter 2. However, the problem with these structures is that each one of them has a pair of sequences either as input or as output. To enhance employing these structures in a stream cipher, we seek a transformation with a single input and a single output. In the following, we introduce the modified DTWT and its inverse which both take a single input and generate a single output.

5.2.1 Modified Wavelet Transform

We propose the modified DTWT and its inverse as in Figure 5.6. It can be verified that the modified DTWT in Figure 5.6a consists of an ordinary DTWT followed by a multiplexer that combines the two sequences w_0 and w_1 without losing information. Assuming the input to the modified DTWT is a one-sided sequence, the input-output relationship is

$$y(n) = \sum_{\ell=0}^{\infty} \mathsf{G}(n,\ell)\, x(\ell) \qquad \forall n \in \mathbb{Z}_{\geq 0} \ , \tag{5.7}$$



(a) Transform.

(b) Inverse transform.

Figure 5.6: Modified DTWT and its inverse.

where

$$\mathsf{G}(n,\ell) := \begin{cases} g_0(n-\ell) & n \text{ even} \\ g_1(n-\ell-1) & n \text{ odd} \end{cases} \qquad \forall n,\ell \in \mathbb{Z}_{\geq 0} \tag{5.8}$$

is a finite sequence called the *kernel* of the modified DTWT. Since sequences g_0 and g_1 are one sided, the following properties of G can be easily deduced using its definition in (5.8).

Causality: $\mathsf{G}(n,\ell) = 0 \quad \forall n,\ell \in \mathbb{Z}_{\geq 0}$ such that $\ell > n$.

Odd Indices: $\mathsf{G}(n,n) = g_1(-1) = 0 \quad \forall n \in \mathbb{N}_{\mathsf{odd}}$.

Periodicity: $\mathsf{G}(n+2,\ell+2) = \mathsf{G}(n,\ell) \quad \forall n,\ell \in \mathbb{Z}_{\geq 0}$.

Coefficients of $\mathsf{G}(n,\ell)$ are listed in Table 5.1 for even and odd time indices n_{even} and n_{odd} in terms of the coefficients of the filters g_0 and g_1. By this table and Fact 2.1.1, we can state the following facts.

FACT 5.2.1. *The kernel $\mathsf{G}(n,\ell)$ in (5.8) has at most $2(L+1)$ nonzero coefficients. Moreover, it is completely specified with $L+1$ finite-field elements.*

FACT 5.2.2. *The convolution in (5.7) can be calculated over the finite support*

$$\mathcal{S}(n) := \begin{cases} \{\, 0,\ldots,n-\pi_n \,\} & n \leq L \\ \{\, n-\pi_n-L,\ldots,n-\pi_n \,\} & n > L \ , \end{cases} \tag{5.9}$$

where $\pi_n := n \mod 2$ is the parity of the time index n.

By this fact, convolution (5.7) can be calculated using a finite memory of length $L+1$.

Table 5.1: Coefficients of $\mathsf{G}(n,\ell)$.

ℓ	$n-L-1$	$n-L$	$n-L+1$	$\cdots$	$n-2$	$n-1$	n
$\mathsf{G}(n_{\mathsf{even}},\ell)$	0	$g_0(L)$	$g_0(L-1)$	$\cdots$	$g_0(2)$	$g_0(1)$	$g_0(0)$
$\mathsf{G}(n_{\mathsf{odd}},\ell)$	$g_1(L)$	$g_1(L-1)$	$g_1(L-2)$	$\cdots$	$g_1(1)$	$g_1(0)$	0

Similar to (5.7), a convolution equation specifies the input-output relationship of the modified inverse DTWT in Figure 5.6b. This equation is

$$y(n) = \sum_{\ell=0}^{\infty} \mathsf{H}(n,\ell)\, x(\ell) \qquad \forall n \in \mathbb{Z}_{\geq 0} \ , \tag{5.10}$$

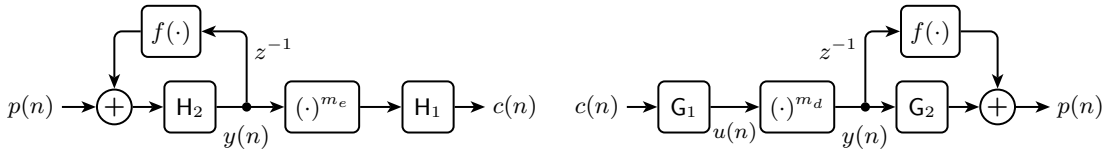
where

$$\mathsf{H}(n,\ell) := \begin{cases} h_0(n-\ell) & \ell \text{ even} \\ h_1(n-\ell-1) & \ell \text{ odd} \end{cases} \tag{5.11}$$

is the kernel. The convolution in (5.10) can be calculated over the finite support $\mathcal{S}(n)$ defined in (5.9). It can be easily verified that H in (5.11) possesses the same properties as G.

5.2.2 Basic Round of the WSSC

In this section, we integrate the modified DTWT and its inverse, proposed in the previous section, with some nonlinear mappings to design a SSC as in Figure 5.7. We propose the structure in Figure 5.7a as the basic round of the encryption system for the WSSC (we recall that the proposed cryptosystem is iterative). In this structure, all the sequences belong to the vector space $\mathbb{F}^*$, where we choose $\mathbb{F} = \mathsf{GF}(256)$ for practical purposes. The blocks H_1 and H_2 are kernels of two modified inverse DTWTs as in (5.11). The mapping $(\cdot)^{m_e} : \mathbb{F} \to \mathbb{F}$ operates on its input sequence symbol-by-symbol, and $m_e \in \mathbb{N}$ is selected such that the mapping is invertible. The function $f : \mathbb{F} \to \mathbb{F}$ is a nonlinear mapping that operates on its input sequence symbol-by-symbol. Inspired by the design of the S-box in



(a) Encryption.

(b) Decryption.

Figure 5.7: Basic round of the WSSC.

advanced encryption standard (AES) [48, 148], we suggest using the extended inversion function for this mapping defined as

$$f(x) := \begin{cases} x^{-1} & x \neq 0 \\ 0 & x = 0 \end{cases} \qquad \forall x \in \mathbb{F} \ . \tag{5.12}$$

In Figure 5.7, the sequences $p \in \mathbb{F}^*$ and $c \in \mathbb{F}^*$ are the plaintext and the ciphertext, respectively. The equations relating the ciphertext to the plaintext in the encryption system of Figure 5.7a are

$$y(n) = \sum_{\ell=0}^{\infty} \mathsf{H}_2(n,\ell) \left[p(\ell) + f\Big(v(\ell-1)\Big) \right] \tag{5.13a}$$

$$c(n) = \sum_{\ell=0}^{\infty} \mathsf{H}_1(n,\ell)\, y^{m_e}(\ell) \tag{5.13b}$$

for all $n \in \mathbb{Z}_{\geq 0}$. The initial state of the first recursive equation is $v(-1) = 0$.

The decryption system of the WSSC (Figure 5.7b) is designed along the guidelines employed in the design of the system in Figure 5.4b. In the structure depicted in Figure 5.7b, G_1 and G_2 are two modified DTWTs as in (5.8). They are designed such that for both $i = 0, 1$, G_i is the inverse of H_i in the encryption system. The mapping $(\cdot)^{m_d} : \mathbb{F} \to \mathbb{F}$ operates symbol-by-symbol on its input sequence. It is the inverse of the mapping $(\cdot)^{m_e}$ in the encryption system. To serve this purpose, we must have $m_e\, m_d \equiv 1 \mod (|\mathbb{F}| - 1)$. The equations describing the decryption system are

$$u(n) = \sum_{\ell=0}^{\infty} \mathsf{G}_1(n,\ell)\, c(\ell) \tag{5.14a}$$

$$p(n) = \sum_{\ell=0}^{\infty} \mathsf{G}_2(n,\ell)\, u^{m_d}(\ell) + f\left(\big[u(n-1)\big]^{m_d} \right) \tag{5.14b}$$

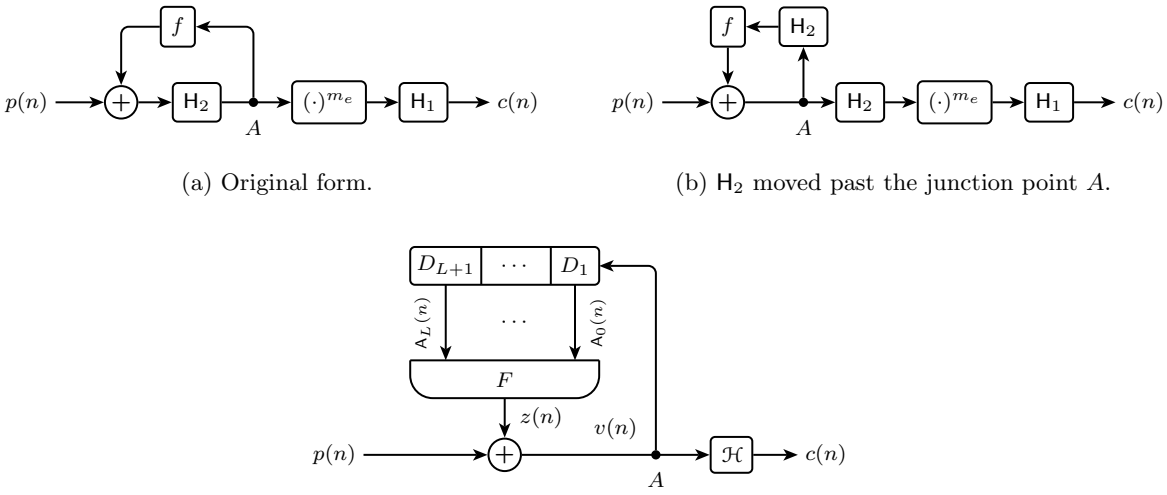
for all $n \in \mathbb{Z}_{\geq 0}$. The initial state in the second recursive equation is $u(-1) = 0$.

As we will explain in Section 5.3, chosen-ciphertext attack is the strongest attack on SSCs. This implies that the design of the decryption system requires careful attention. Therefore, we choose the exponent m_d such that the power mapping $(\cdot)^{m_d}$ possesses the highest possible nonlinearity. The (non)linearity of power mappings over fields of characteristic two is measured using the Walsh transform [149, 69]. To have a one-to-one power

mapping $(\cdot)^{m_d}$ with the highest possible nonlinearity over $\mathsf{GF}(2^r)$, we select $m_d = 2^{(r/2)+1}-1$ in the case $r \equiv 0 \mod 4$ [69]. Since the underlying field in our design is $\mathsf{GF}(2^8)$, we set $m_d = 31$. For the this value, we obtain $m_e = m_d^{-1} \mod 255 = 181$.

To reveal the relationship between one round encryption of the WSSC and the canonical form of all SSC in Figure 5.4a, we transform the block diagram in Figure 5.7a. The transformation process, depicted in Figure 5.8, consists of two steps. Figure 5.8a shows the original form of the encryption block diagram[1]. The first step is moving the H_2 transform past the junction point A after which we get the configuration in Figure 5.8b. Since H_2 is, in fact, a filtering operation with time-dependent coefficients, it can be virtually implemented using a shift register. The length of this register is $L+1$ (similar to Fact 5.2.1). Hence, the final step is reconfiguration to the structure in Figure 5.8c in which $F : \mathbb{F}^{L+1} \to \mathbb{F}$ is a nonlinear function defined as $F(x_0, \ldots, x_L) := f(\sum_{i=0}^{L} x_i)$, for all $x_0, \ldots, x_L \in \mathbb{F}$, and $\mathsf{A}_0(n), \ldots, \mathsf{A}_L(n)$ are the coefficients of the kernel H_2 defined as $\mathsf{A}_\ell(n) := \mathsf{H}_2(n, n - \pi_n - \ell)$, for all $\ell \in (L+1)$, where $\pi_n := n \mod 2$ is the parity of the time index n. With this

[1]The delay in the feedback path is omitted only to simplify figures.



(a) Original form.

(b) H_2 moved past the junction point A.

(c) Canonical form.

Figure 5.8: Transforming the WSSC encryption to the canonical form of SSCs in Figure 5.4a.

notation, at any time instance $n \in \mathbb{Z}_{\geq 0}$, the output of the function F is

$$z(n) = f\left(\sum_{\ell=0}^{L} \mathsf{A}_\ell(n)\, v(n - \pi_n - 1 - \ell)\right) \quad . \tag{5.15}$$

In Figure 5.8c, $\mathcal{H}$ is the cascade of the three transformations H_2, $(\cdot)^{m_e}$, and H_1 in the mentioned order.

A comparison between the canonical model of Figure 5.4a and the modified structure in Figure 5.8c reveals similarities, in spite of which, we emphasize the following main differences.

1. As explained before, in the design of SSCs, usually LFSRs are employed to realize FSMs. However, in the WSSC, the FSM is implemented using only a shift register without a feedback.

2. The secret key in WSSC determines the time-dependent weights with which the contents of the shift register are combined whereas in an SSC it determines the initial state of the LFSR.

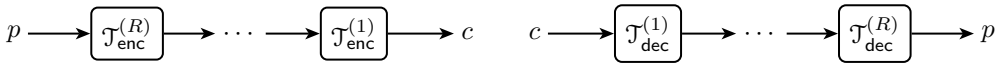
By a similar approach, one shows that the decryption transform of the WSSC in Figure 5.7b consists of a sequence transform followed by the decryption system of an SSC.

In the following, we study the proposed WSSC in the multiple rounds configuration.

5.2.3 Multiple Rounds of the WSSC

To increase the security of the WSSC, we can cascade multiple rounds of the encryption and the decryption systems depicted in Figure 5.7. Assuming R is the number of rounds, let $\mathcal{T}_{\mathsf{enc}}^{(i)} : \mathbb{F}^* \to \mathbb{F}^*$ be the transfer function of the encryption system, described by (5.13), for the i-th round. In addition, let $\mathcal{T}_{\mathsf{dec}}^{(i)} : \mathbb{F}^* \to \mathbb{F}^*$ be the corresponding decryption transfer function described by (5.14) for $i = 1, \ldots, R$. The encryption and decryption systems in multiple round arrangements are shown in Figure 5.9. As shown in the figure, the encryption and decryption blocks are arranged in the reverse order in the cascade chains.

Every modified DTWT, as explained in Section 5.2.1, is realizable using $L + 1$ memory units each of which holds only one element of $\mathbb{F}$. Since every encryption round consists of



(a) Encryption. (b) Decryption.

Figure 5.9: Multiple rounds of the WSSC.

two modified DTWTs, the total amount of memory required to implement the encryption chain in Figure 5.9a is $2R(L+1)$. The same amount of memory is required to implement the decryption chain in Figure 5.9b. Therefore, the security gained by increasing the number of rounds increases the delay in both the encryption and the decryption cascades. Precisely, after processing a long plaintext or ciphertext sequence, the user has to wait for $2R(L+1)$ time units to flush out the system memory. In the remainder of this section, we explain how the secret key is involved in the specification of the WSSC.

5.2.4 Key Setup

To use multiple rounds of the WSSC, all the components of the encryption and the decryption systems must be specified. The nonlinear function f, defined in (5.12), is the same for all rounds. Therefore, the modified DTWTs and their inverses are the only components that have to be specified based on the secret key. As explained in Section 5.2.1, a DTWT and its inverse are designed based on a PU matrix. Therefore, the secret key determines the parameters of the PU building blocks.

Let $\mathsf{G}_1^{(i)}$, $\mathsf{G}_2^{(i)}$, $\mathsf{H}_1^{(i)}$, and $\mathsf{H}_2^{(i)}$ be the modified DTWTs and their inverses employed in the i-th round for $i = 1, \ldots, R$. For every round of the WSSC, the parameters determining the modified DTWTs and their inverses are listed in Table 5.2. In this list, only the finite-field parameters of the PU building blocks are kept secret and determined based on the secret key. The rest of them are made public[2]. By this table, a secret key of length $\kappa = \beta_1 + \beta_2$ suffices to generate a PU matrix, using Algorithm 2.2, which is used to design a modified DTWT transform-inverse pair.

[2]We note that it is possible to make all these parameters key dependent.

Table 5.2: Design parameters for every round of WSSC.

Public	$\beta_1 \in \mathbb{Z}_{\geq 0}$	Number of degree-one building blocks
	$\beta_2 \in \mathbb{Z}_{\geq 0}$	Number of degree-2τ building blocks
	$\boldsymbol{\tau} \in \mathbb{N}^{\beta_2}$	Degrees of the degree-2τ building blocks
	$\boldsymbol{\mu} \in \{\,1,2\,\}^{\beta_1+\beta_2}$	Order in which the PU building blocks are multiplied
Secret	$\mathbf{u} \in \mathbb{F}^{\beta_1}$	Design parameters for the degree-one building blocks
	$\boldsymbol{\zeta} \in \mathbb{F}^{\beta_2}$	Design parameters for the degree-2τ building blocks

To design a WSSC consisting of R rounds, $2R$ modified DTWT transform-inverse pairs are required. One of them is designed using the secret key. The rest are designed using secret vectors obtained by expanding the secret key vector. For this purpose, we employ Algorithm 5.2, with $\epsilon = 2R$, to expand the secret key. This algorithm takes the secret key $\mathbf{k} \xleftarrow{\$} \mathbb{F}^{\kappa}$ as the input and outputs the set $\mathcal{K}$ consisting of $2R$ vectors of the same length. The structure of the key-expansion algorithm is very similar to that employed in the block cipher AES [48]. The design criteria, similarly, is having nonlinear relations between each output vector and the master key such that taking advantage of these relations in an attack is infeasible. In Algorithm 5.2, the binary operation $\oplus$ is the bitwise exclusive-OR and

Algorithm 5.2: Key expansion.

Input: Master key $\mathbf{k} = [\,k_1, \ldots, k_\kappa\,]^\dagger \in \mathbb{F}^{\kappa}$

Output: Parameter set $\mathcal{K} = \{\,\mathbf{k}_1, \ldots, \mathbf{k}_T\,\} \subset \mathbb{F}^{\kappa}$

1. **for** $i = 1$ **to** κ **do** $k_{i1} \leftarrow k_i$
2. $\mathbf{k}_1 \leftarrow [\,k_{11}, \ldots, k_{\kappa 1}\,]^\dagger$
3. **for** $j = 2$ **to** T **do**
4. $\quad k_{1j} \leftarrow k_{1,j-1} \oplus k_{1,j-1}^{-1} \oplus c_{j-1}$
5. $\quad$ **for** $i = 2$ **to** κ **do** $k_{ij} \leftarrow k_{i,j-1} \oplus k_{i-1,j}^{-1}$
6. $\quad \mathbf{k}_j \leftarrow [\,k_{1j}, \ldots, k_{\kappa j}\,]^\dagger$
7. **end**

$c_1, \ldots, c_{2R-1}$ are public constants.

To find the relationship between $L = \mathsf{Deg}_{z^{-1}} G_0(z) = \mathsf{Deg}_{z^{-1}} G_1(z)$ and the length of the secret key κ, we first calculate the order of the PU matrix $\mathbf{P}(z)$. We note that the degree and the order of a matrix polynomial are not necessarily equal [186]. Nevertheless, the order is always greater than or equal to the degree. Therefore, $\mathsf{Ord}_{z^{-1}} \mathbf{P} \geq \beta_1 + \sum_{i=1}^{\beta_2} \tau_i$, where $\tau_1, \ldots, \tau_{\beta_2}$ are the parameters of the $\mathbf{S}$ building blocks. By the polyphase representations (2.5), the parameter L is related to $\mathsf{Ord}_{z^{-1}} \mathbf{P}$ as $L = 2\mathsf{Ord}_{z^{-1}} \mathbf{P} + 1$. Combining these two results, we get

$$L \geq 2 \left(\beta_1 + \sum_{i=1}^{\beta_2} \tau_i \right) + 1 \ . \tag{5.16}$$

The typical length of the secret key is $\kappa = 16$ $\mathsf{GF}(256)$-elements that is equivalent to 128 bits. To calculate a typical value for L, we consider the scenario in which $\beta_1 = \beta_2 = 8$ and, hence, $\kappa = 16$. We assume all $\mathbf{S}$ building blocks have degree two. By (5.16), we have $L \geq 33$.

5.3 Cryptanalysis of the WSSC

In this section, we study the security of the proposed WSSC with respect to some known and relevant cryptanalytic techniques and also present a new method specific to our stream cipher. As Maurer has shown in [134], the chosen-ciphertext attack is the most powerful attack against SSCs. Therefore, in this section, we mostly focus our attacks on the decryption system of Figure 5.7b.

Attacks studied in this section are: interpolation, algebraic (such as Gröbner basis and XL algorithm), delta, time-memory tradeoff, divide and conquer, correlation, and distinguishing. Among them, the delta attack is specifically developed for the WSSC. We will show that one round of the system is vulnerable to this attack. We propose two methods to prevent it. The first one is increasing the number of rounds. The other one is employing combiners with memory. We highly recommend using at least two rounds of the WSSC.

5.3.1 Interpolation Attack

Cryptosystems with algebraic relations between the plaintext and the ciphertext might be vulnerable to this attack [108, 146, 107]. In the interpolation attack attack, the adversary establishes an algebraic relation between the plaintext and the ciphertext with unknown coefficients. Using a set of plaintext-ciphertext pairs, the adversary is able to construct a system of linear equations in the unknown coefficients. After solving this linear system, the adversary can decrypt any ciphertext without knowing the key.

By (5.14), every plaintext symbol is a function of the current and a limited number of previous ciphertext symbols. To study the feasibility of applying the interpolation attack to the WSSC, let $\gamma := p(n)$, $\chi_i := c(n-i)$, and $\omega_i := u(n-i)$, for all $i = 0, \ldots, n$, at an arbitrary and fixed time instance $n \in \mathbb{Z}_{\geq 0}$. We first study the case in which n is a large enough and even integer. By (5.14a) and (5.9), we have $\omega_j = \sum_{i=0}^{L} a_{ij}\, \chi_{i+j}$, where $a_{ij} := \mathsf{G}_1(n-j, n-j-i)$. Hence, $\omega_j^{m_d}$ is a homogenous polynomial of degree m_d in $\chi_j, \ldots, \chi_{j+L}$. To count the number of terms in this polynomial, let w be the Hamming weight of m_d, i.e., $m_d = 2^{\theta_1} + \cdots + 2^{\theta_{\mathsf{w}}}$ for proper $\theta_1, \ldots, \theta_{\mathsf{w}} \in \mathbb{Z}_{\geq 0}$. Since for any $\theta \in \mathbb{Z}_{\geq 0}$, the power function $(\cdot)^{2^\theta}$ is $\mathbb{F}$-linear, we have

$$\omega_j^{m_d} = \prod_{\ell=1}^{\mathsf{w}} \left(\sum_{i=0}^{L} \chi_i^{2^{\theta_\ell}} \right) \quad . \tag{5.17}$$

From here, we deduce that $\omega_j^{m_d}$ has $(L+1)^{\mathsf{w}}$ terms.

Now, by (5.14b), we have $\gamma = \sum_{j=0}^{L} b_j\, \omega_j^{m_d} + \omega_1^{255-m_d}$, where $b_j := \mathsf{G}_2(n, n-j)$, and we have used the equality $\chi^{-m_d} \equiv \chi^{255-m_d}$ over $\mathsf{GF}(256)$. The summation expression of γ is a homogenous polynomial of degree m_d in $\chi_0, \ldots, \chi_{2L}$ with maximum $(L+1)^{\mathsf{w}+1}$ terms. The term $\omega_1^{255-m_d}$ is a homogenous polynomial of degree $255 - m_d$ in $\chi_1, \ldots, \chi_{L+1}$ that has $(L+1)^{\mathsf{w}'}$ terms, where w' is the Hamming weight of $255 - m_d$.

All results obtained for even n can be easily translated to the case in which n is odd. One of the differences is that, by (5.9), ω_j is a polynomial in $\chi_{j+1}, \ldots, \chi_{j+L+1}$. In addition, γ is a polynomial in $\chi_1, \ldots, \chi_{2L+1}$. Collectively, we have proved the following lemma.

LEMMA 5.3.1. *At an arbitrary and fixed time instance $n \in \mathbb{Z}_{\geq 0}$, the plaintext symbol $p(n)$*

is a polynomial in $2L+1$ *ciphertext symbols. It consists of two homogenous components of degrees* m_d *and* $255-m_d$*. Assuming* w *and* w' *are respectively the Hamming weights of these degrees,* $p(n)$ *has* $(L+1)^{\mathsf{w}+1}+(L+1)^{\mathsf{w}'}$ *terms.*

For $m_d = 31$, we have $\mathsf{w} = 5$ and $\mathsf{w}' = 3$. For the typical value $L = 33$, the maximum number of terms is approximately 2^{31}. Solving a system of linear equations for 2^{31} unknowns has complexity $O(2^{93})$ even using fast algorithms such as LUP decomposition [40]. This complexity is even higher than that in the time-memory tradeoff attack that we will explain later in this section. Hence, the interpolation attack is infeasible on one round of the WSSC. For two rounds, the described version of the interpolation attack is trivially infeasible too.

Another version of the interpolation attack is meet-in-the-middle attack [108]. In this version, the adversary writes an arbitrarily selected sequence in the middle of the system as algebraic equations of both the input and the output sequences. After equating these two equations and eliminating the middle sequence, she establishes a system of linear equations from which the unknowns are obtained. The advantage of this version of the attack is that the equations might be simpler in some cases. Assume this attack is mounted on the decryption system of Figure 5.7b with respect to the sequence $y(n)$. The two equations relating $y(n)$ to the input and output sequence respectively are:

$$y(n) = \left[\sum_{\ell\in\mathcal{S}(n)} \mathsf{G}_1(n,\ell)\, c(\ell)\right]^{m_d} \tag{5.18a}$$

$$y(n) = \sum_{\ell\in\mathcal{S}(n)} \mathsf{H}_2(n,\ell)\left[p(\ell) + f\Big(y(\ell-1)\Big)\right] \ . \tag{5.18b}$$

In writing (5.18b), we have used the encryption system in Figure 5.7a and the fact that $(\mathsf{G}_1, \mathsf{H}_1)$ is a transform-inverse pair. Equating the two equations in (5.18) does not eliminate $y(n)$. As a matter of fact, the result is a multivariate polynomial in $y(0), \ldots, y(n-1)$ besides plaintext and ciphertext symbols as its variables. Since the number of variables increases by time, it is not possible to easily recover plaintext symbols. Therefore, this version of the interpolation attack is more complicated than its original form. This observation renders the meet-in-the-middle attack infeasible on one round of the WSSC. For similar reasons,

the described attack is also infeasible on two rounds of the system.

5.3.2 Algebraic Attacks

The goal in the interpolation attack is constructing a system of linear equations because such systems, if of moderate size, can be efficiently solved. However, we showed that the system describing every plaintext symbol in terms of the ciphertext symbols in one round of the WSSC is too large to be feasibly solved. We note that the unknown coefficients obtained by an interpolation attack are related to each other through their dependencies to the coefficients of the kernels G_1 and G_2. The interpolation attack ignores these dependencies to construct linear equations. To reduce the number of unknowns and, hence, the size of the system, we rewrite these equations in terms of the $\mathsf{G}_1(n,\ell)$ and $\mathsf{G}_2(n,\ell)$ coefficients. In contrast to the interpolation attack, the resulting equations are nonlinear. The total number of unknowns is equal to the number of coefficients required to specify these kernels. By Fact 5.2.1, each of the two kernels is completely specified with $L+1$ coefficients. In the following, we study different methods for solving systems of multivariate polynomial equations.

A Gröbner Basis

This is a general tool for solving all systems of polynomial equations [14, 73, 45, 46], which is a generalization of the Gaussian elimination method. To solve the system

$$\begin{cases} f_1(x_1,\ldots,x_\nu)=0 \\ \qquad\vdots \\ f_m(x_1,\ldots,x_\nu)=0 \end{cases} \tag{5.19}$$

over a field $\mathbb{F}$, one obtains the Gröbner basis of the ideal $I=(f_1,\ldots,f_m)$ with respect to the reverse lexicographic ordering. If $\{\, g_1,\ldots,g_{m'} \,\}$ is a Gröbner basis of I, in which $m' \le \min\{\, m,\nu \,\}$, then the polynomial g_i depends only on the variables $x_i,\ldots,x_\nu$ for all $i=1,\ldots,m'$. If $m'=\nu$, then $g_{m'}$ is a univariate polynomial in x_ν, which its root can be solved using efficient root finding algorithms [125, 82]. The value of x_ν obtained from here

is substituted in $g_{m'-1}$ that is a polynomial in $x_{\nu-1}$ and x_ν. Solving the resulting univariate polynomial gives the value of $x_{\nu-1}$. Continuing this procedure, one obtains all unknowns. If $m' < \nu$, then $g_{m'}$ is a multivariate polynomial in $x_{m'}, \ldots, x_\nu$. By randomly assigning values to $x_{m'+1}, \ldots, x_\nu$, we turn $g_{m'}$ into a univariate polynomial in $x_{m'}$ that its roots are efficiently obtained. The rest of the root finding method is similar to the previous case.

The complexity of all algorithms for computing the Gröbner basis of a random system of polynomial equations is exponential in the number of variables [73]. However, in cases where either the number of variables or the degree of polynomials is too small, fast algorithms such as the F_4 [78] or the F_5 [79] algorithm might be efficiently applicable. Using an approach similar to the one used in the interpolation attack, one can prove the following fact.

FACT 5.3.1. *The equation governing one decryption round of the WSSC is a polynomial in* $\nu = 2(L+1)$ *variables consisting of two homogenous parts of degrees* $m_d + 1$ *and* $255 - m_d$.

For the choices $L = 33$ and $m_d = 31$, the number of variables and the total degree are 68 and 224, respectively. Computing the Gröbner basis of a system of equations with these specifications seems to be infeasible [44].

B XL Algorithm

The XL algorithm is an alternative method for solving over-defined systems of homogenous polynomial equations [44, 203]. To briefly explain this method, consider the system of polynomial equations (5.19) in which all polynomials are assumed to be homogenous of degree $\theta \in \mathbb{N}$. The main ideas of the XL algorithm and the method of Gröbner basis are essentially the same: constructing the ideal $I = (f_1, \ldots, f_m)$ and finding a set of polynomials $g_1, \ldots, g_{m'} \in I$ such that solving the system $g_1 = \cdots = g_{m'} = 0$ is easier than solving the original system [11]. In the XL algorithm, the new system is constructed by polynomials of the form $(\prod_{j \in \mathcal{J}} x_j)\, f_i$, where $\mathcal{J} \subset \{1, \ldots, \nu\}$ such that $|\mathcal{J}| = D - \theta$ for some fixed $D \in \mathbb{N}_{\geq \theta}$ (i.e., D is the total degree of the new equations). In every equation of the new system, each term with total degree less than or equal to D is replaced by a new variable. This step is called *linearization* that was, first, used in [116] to break the HFE public-key cryptosystem

[155]. This technique reduces the total degree of the system of equations and hence reduces the complexity of inverting it.

To obtain the complexity of the XL algorithm in our case, we follow the guidelines of [44] for the complexity calculation. Given m original homogenous equations of degree θ in ν variables, the number of generated equations is $N_{\mathsf{eq}} = \frac{\nu^{D-\theta}}{(D-\theta)!} \cdot m$ while we have about $N_{\mathsf{var}} = \frac{\nu^D}{D!}$ linear variables. The method is successful when $N_{\mathsf{eq}} \geq N_{\mathsf{var}}$, i.e., when

$$m \geq \frac{\nu^\theta}{D(D-1)\cdots(D-\theta+1)} = O(\nu^\theta) \ . \tag{5.20}$$

By Fact 5.3.1, the one-round decryption equation of the WSSC is a polynomial in $\nu = 2(L+1)$ variables consisting of two homogenous parts of degrees m_d and $255-m_d$. With the typical values $L = 33$ and $m_d = 31$, we have $\nu = 68$ and $\theta = \max\{\, m_d,\, 255 - m_d \,\} = 224$. By (5.20), the minimum number of required equations is in the order of 2^{1364}. The eventual system of linear equations with this huge size is completely infeasible to solve practically.

C FXL Algorithm

The FXL is an extension of the XL algorithm in which the values of some variables are randomly guessed and the resulting system is solved [44]. The random guessing technique is intended to make the system of equations over-defined specially when $m \approx \nu$. The final system of linear equations in the FXL is expected to be much smaller than that in the XL. As explained in [44], there is a tradeoff between the size of search space (determined by the number of guesses) and the complexity of the eventual linear system. The number of guesses recommended in [44] is $O(\sqrt{\nu})$ in which case the complexity of the FXL approximately becomes $2^{r\sqrt{\nu}(\log_2 \nu+8)}$ over $\mathsf{GF}(256)$. Here, $r = m-\nu \geq 2$ is the number of excess equations. With the value $\nu = 68$ in WSSC, the complexity of FXL becomes at least 2^{232}, which is still out of the computational reach of today computers.

5.3.3 Delta Attack

This is a chosen-ciphertext attack in which a number of ciphertext streams with a special form are chosen and applied to the decryption system of the WSSC in Figure 5.7b. Using

the corresponding plaintext streams, it is computationally possible to recover the coefficients of $\mathsf{G}_1(n,\ell)$ and $\mathsf{G}_2(n,\ell)$ in only one round of the WSSC. The ciphertext streams employed in this attack are of the form

$$c_{n_0}(n;\,\alpha) := \begin{cases} \alpha & n = n_0 \\ 0 & n \neq n_0 \end{cases} \tag{5.21}$$

in which $n_0 \in \{\,0,1\,\}$ and $\alpha \in \mathbb{F} \setminus \{\,0\,\}$. Let $p_{n_0}(n;\,\alpha)$ be the corresponding plaintext stream. In the following, we explain this attack for one and two rounds. We will discuss that this attack is unsuccessful on two rounds.

A One Round

By (5.14a), we have $u(n) = \alpha\,\mathsf{G}_1(n,n_0)$. Substituting this result in (5.14b), we get

$$p_{n_0}(n;\,\alpha) = \alpha^{m_d}\,\mathsf{X}(n,n_0) + f(\alpha^{m_d}\,[\mathsf{G}_1(n-1,n_0)]^{m_d}) \ , \tag{5.22}$$

where

$$\mathsf{X}(n,n_0) := \sum_{\ell\in\mathcal{S}(n)} \mathsf{G}_2(n,\ell)\,[\mathsf{G}_1(\ell,n_0)]^{m_d} \ . \tag{5.23}$$

Using another ciphertext-plaintext pair $(c_{n_0}(n;\,\beta),\,p_{n_0}(n;\,\beta))$ with $\beta \neq \alpha$, we obtain a system of equations. After eliminating $\mathsf{X}(n,n_0)$ in this system, we get the polynomial equation

$$\alpha^{m_d}\,f\Big(\beta^{m_d}\,[\mathsf{G}_1(n-1,n_0)]^{m_d}\Big) + \beta^{m_d}\,f\Big(\alpha^{m_d}\,[\mathsf{G}_1(n-1,n_0)]^{m_d}\Big) + \alpha^{m_d}\,p_{n_0}(n;\,\beta) + \beta^{m_d}\,p_{n_0}(n;\,\alpha) = 0 \ . \tag{5.24}$$

For the extended inversion function f in (5.12), this polynomial equation has the unique solution

$$\mathsf{G}_1(n,n_0) = \left[\frac{(\alpha/\beta)^{m_d} + (\beta/\alpha)^{m_d}}{\alpha^{m_d}\,p_{n_0}(n+1;\,\beta) + \beta^{m_d}\,p_{n_0}(n+1;\,\alpha)}\right]^{m_e} \ . \tag{5.25}$$

Therefore, only two ciphertext streams as in (5.21) with $n_0 = 0,1$ suffice to completely find the coefficients of G_1. Once G_1 is known, we obtain $\mathsf{X}(n,n_0)$ from (5.22).

$$\mathsf{X}(n,n_0) = \tfrac{1}{\alpha^{m_d}}\left[p_{n_0}(n;\,\alpha) + f(\alpha^{m_d}\,[\mathsf{G}_1(n-1,n_0)]^{m_d})\right] \tag{5.26}$$

The coefficients of G_2 can be obtained from the definition of X in (5.23).

B Two Rounds

Consider two decryption rounds of the WSSC in Figure 5.10. Streams at different points of this system in response to the ciphertext stream in (5.21) are:

$$u(n) = \alpha\, \mathsf{G}_1^{(1)}(n, n_0) \ , \tag{5.27a}$$

$$v(n) = \alpha^{m_d} \underbrace{\sum_{\ell \in \mathcal{S}(n)} \mathsf{G}_2^{(1)}(n, \ell) \left[\mathsf{G}_1^{(1)}(\ell, n_0)\right]^{m_d}}_{\mathsf{X}(n, n_0)} + \alpha^{-m_d} \left[\mathsf{G}_1^{(1)}(n-1, n_0)\right]^{-m_d} \ , \tag{5.27b}$$

$$y(n) = \alpha^{m_d} \underbrace{\sum_{\ell \in \mathcal{S}(n)} \mathsf{G}_1^{(2)}(n, \ell) \left[\mathsf{X}(\ell, n_0)\right]^{m_d}}_{\mathsf{Y}(n, n_0)} + \alpha^{-m_d} \underbrace{\sum_{\ell \in \mathcal{S}(n)} \mathsf{G}_1^{(2)}(n, \ell) \left[\mathsf{G}_1^{(1)}(\ell - 1, n_0)\right]^{-m_d}}_{\mathsf{Z}(n, n_0)} \ , \tag{5.27c}$$

and

$$\begin{aligned} p_{n_0}(n;\, \alpha) = \sum_{\ell \in \mathcal{S}(n)} \mathsf{G}_2^{(2)}(n, \ell) \left[\alpha^{m_d}\, \mathsf{Y}(\ell, n_0) + \alpha^{-m_d}\, \mathsf{Z}(\ell, n_0)\right]^{m_d} \\ + \left[\alpha^{m_d}\, \mathsf{Y}(n-1, n_0) + \alpha^{-m_d}\, \mathsf{Z}(n-1, n_0)\right]^{-m_d} \end{aligned} \tag{5.27d}$$

Staring from $n = 0$ and increasing n by one at a time, the adversary can find a few first coefficients of some kernels. However, as n grows large, the equations become highly complicated with many terms. The stream observed by the adversary $p_{n_0}(n;\, \alpha)$ consists of highly nonlinear terms generated by the power mapping $(\cdot)^{m_d}$ in the design of the decryption system in Figure 5.7b. Therefore, linear approximations of these terms are very unlikely to reduce the complexity of solving the final equations for the kernels coefficients. Hence, the delta attack fails on two rounds of the system. We note that in practice, similar to RC4, one can ignore a few first output symbols to prevent revealing any kernel coefficients.

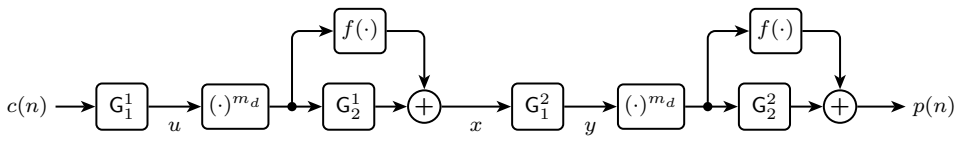


Figure 5.10: Two decryption rounds of the WSSC.

C Discussion

The modified DTWTs H and G in the design of the WSSC help to diffuse the symbols of the input stream in the output. Every one of them, which consists of two one-sided filters (h's and g's) as in Figure 5.6, spreads the length of its input stream by a factor proportional to its length. Precisely, if $\mathcal{L}_{\mathsf{in}}$ and $\mathcal{L}_{\mathsf{out}}$ are the lengths of input and output streams of H or G, then we have the relationship $\mathcal{L}_{\mathsf{out}} = \mathcal{L}_{\mathsf{in}} + L$, where $L+1$ is the maximum length of the h and g filters. A delta ciphertext stream as in (5.21) delays the dissipation of its only coefficient in the output. In one decryption round of the system, the diffusion of the delta stream does not reached to a point where the final equations are complicated enough. Therefore, at least two rounds of the WSSC are required to resists the delta attack. We note that replacing the nonlinear function f, in the design of the WSSC, with a highly complicated S-box does not improve the resistance to the delta attack. This is because every function f has a polynomial representation over $\mathbb{F}$ by Lagrange's interpolation formula. This representation yields the univariate polynomial equation (5.24) that can be efficiently solved.

One approach to prevent the delta attack if a single round of the system is used is to employ combiners with memory for the nonlinear function f [170, 10]. In the case of WSSC, we suggest using a very simple combiner $f : \mathbb{F}^M \to \mathbb{F}$ that takes as input the current and past $M-1$ symbols of the input stream. If y and z are the input and output streams of f, respectively, then $z(n) = f(y(n), \ldots, y(n-M+1))$. In fact, we add a shift register of length M at the input of the function f in our previous design. We assume the initial state of this register is zero. This combiner makes the polynomial (5.24) multivariate, which requires other similar equations to construct a system. As we discussed before, such system of nonlinear polynomial equations cannot be easily solved. We note that the adversary might be able to compute only a few first coefficients of G_1 by observing the output stream at time $n = 1$. However, as the time progresses, the complexity of the polynomial equation (5.24) increases.

5.3.4 Time-Memory Tradeoff Attack

Two obvious attacks applicable on all ciphers are exhaustive key search and table pre-computation. In the first one, given a single plaintext-ciphertext pair, the attacker exhaustively tries all possible keys. The matching pair reveals the secret key. The worst-case time complexity of this attack is $T = |\mathbb{F}|^{\kappa} = 2^{8\kappa}$. Considering that typically $\kappa = 16$ in the WSSC, the time complexity of this attack is $T = 2^{128}$, which is infeasible to mount with the state of the art in computational power.

Table pre-computation is the second obvious attack in which, in a chosen-ciphertext attack, the attacker generates a table listing the plaintext sequences corresponding to a fixed ciphertext sequence decrypted under all possible secret keys. Once the pre-computation is over, the attack can be carried out almost instantly. The size of the memory required to store all possible keys is 2^{128} bits that is completely beyond the storage capabilities of today.

In the spectrum covered by the exhaustive key search and the table pre-computation, there are proposals that make a tradeoff between the time complexity and the storage memory [102, 27, 150]. The first one of such methods was proposed by Hellman in [102]. To briefly explain this method, let $\mathsf{Dec}_{\mathbf{k}} : \mathbb{F}^* \to \mathbb{F}^*$ be the decryption function of the WSSC with the secret key $\mathbf{k} \in \mathbb{F}^{\kappa}$. In addition, define the function $\lambda : \mathbb{F}^{\kappa} \to \mathbb{F}^{\kappa}$ as $\mathbf{k} \mapsto \rho\Big(\mathsf{Dec}_{\mathbf{k}}(c_0)\Big)$ for all $\mathbf{k} \in \mathbb{F}^{\kappa}$ and a fixed ciphertext c_0, where ρ is either the identity map or a reduction function. For any $\mathbf{k} \in \mathbb{F}^{\kappa}$, computing $\lambda(\mathbf{k})$ is as simple as decryption. However, inverting $\lambda(\mathbf{k})$ is equivalent to cryptanalysis. Therefore, if WSSC is to be secure, λ must be a OWF. As part of the pre-computation, the cryptanalyst uniformly at random chooses m distinct keys $\mathbf{k}_{10}, \ldots, \mathbf{k}_{m0}$ and computes the chains

$$\begin{array}{c} \mathbf{k}_{10} \xrightarrow{\lambda} \mathbf{k}_{11} \xrightarrow{\lambda} \cdots \xrightarrow{\lambda} \mathbf{k}_{1t} \\ \vdots \\ \mathbf{k}_{m0} \xrightarrow{\lambda} \mathbf{k}_{m1} \xrightarrow{\lambda} \cdots \xrightarrow{\lambda} \mathbf{k}_{mt} \end{array} \tag{5.28}$$

each of length $t \in \mathbb{N}$. To reduce memory requirements, the adversary discards all intermediate points and only stores pairs $(\mathbf{k}_{i0}, \mathbf{k}_{it})$ for all $i \in \{\, 1, \ldots, m \,\}$.

Given a ciphertext-plaintext pair (c_0, p_0) obtained using the secret key $\mathbf{k}$, i.e., $p_0 = \mathsf{Dec}_{\mathbf{k}}(c_0)$, the adversary computes $S_1 = \rho(p_0)$ checks the endpoints of all chains. If $S_1 = \mathbf{k}_{jt}$ for only one $j \in \{1, \ldots, t\}$, then the secret key must be $\mathbf{k}_{j(t-1)}$. Since all the intermediate points are discarded, the adversary obtains the secret key through $t-1$ applications of the function λ on the beginning point $\mathbf{k}_{j0}$. If more than one match is found, i.e., the chains collide at the end, the adversary obtains all possible keys and checks them via decrypting c_0 under each one and comparing the result with p_0. If no match is found, the adversary computes $S_2 = \lambda(S_1)$ and checks if it is an endpoint. If $S_2 = \mathbf{k}_{\ell t}$ for some $\ell \in \{1, \ldots, m\}$, then the secret key is $\mathbf{k}_{\ell(t-2)}$, which is obtained through $t-2$ application of λ on the beginning point $\mathbf{k}_{\ell 0}$. In a similar way, the cryptanalyst computes $S_3 = \lambda(S_2), S_4 = \lambda(S_3), \ldots$, if necessary, and checks the endpoints. If no match is found at all, the adversary has to pre-compute a different table by choosing different initial points and/or the fixed ciphertext c_0. As shown in [102], this method can recover a key in $T^{2/3}$ operations using $T^{2/3}$ words of memory (when $t = m = T^{1/3}$). In the case of the WSSC, we have $T^{2/3} \approx 2^{85.3}$. The time-memory tradeoff attack of [102] with this complexity is infeasible to mount.

A similar time-memory technique is proposed in [27] that employs distinguished points, i.e., points that possess a property that can be easily checked. Complexities of this technique, although lower, are in the same order of those in the previous method. The latest version of time-memory tradeoff attacks is proposed in [150]. This new method improves the efficiency of such attacks by constant factors. Although it presents a threat to ciphers with short keys such as data encryption standard (DES), the proposed WSSC is immune against it by the choice of the key length.

5.3.5 Divide-and-Conquer Attack

The main idea in this attack is dividing the secret information (the secret key or some other related material) into distinct pieces, making guesses for the values of some pieces, and retrieving values of the others using observed information (e.g., the plaintext in the chosen-ciphertext attack). If the bit lengths of the secret information and the pieces respectively

are κ and $\kappa_1, \ldots, \kappa_M$ (such that $\kappa_1 + \cdots + \kappa_M = \kappa$), then the complexity of the divide-and-conquer attack becomes $\sum_{i=1}^{M} 2^{\kappa_i}$ that can be much smaller than the claimed complexity 2^{κ}.

Consider mounting this attack on one decryption round of the WSSC in Figure 5.7b. One approach is guessing the coefficients of the kernel H_1 and solving some equations for the coefficients of H_2. However, by Fact 5.2.1, every one of these kernels is specified with $L + 1$ finite-field elements. Considering that usually $L + 1 > \kappa$ (e.g., typical values are $L = 33$ and $\kappa = 16$), this approach is more complex than the simple exhaustive search. Another approach is using the relationships between the kernel coefficients and the secret key to determine H_1. As we explained in Section 5.2.4, the secret information used to specify any one of the kernels is derived from the secret key through a key expansion algorithm. Therefore, guessing the secret information used to specify H_1 is equivalent to guessing the secret key. Collectively, we conclude that the divide-and-conquer attack is at least as complex as the exhaustive search.

5.3.6 Correlation and Distinguishing Attacks

Proposed in [177], the main idea of the correlation attack is exploiting the correlation between the LFSR output a_n of a stream cipher and the key stream z_n. As shown in Figure 5.4a, the output of the LFSR is passed through a nonlinear Boolean function f to destroy all linear dependencies. However, since Boolean functions are imperfect, there always exist some correlation. The correlation attack converts the problem of recovering the sequence a_n from the the sequence z_n into the following decoding problem. The sequence z_n can be considered a distorted version of a_n after transmitting through a binary symmetric channel that is modeled by f. Since f is publicly known, the channel characteristics are known as well. Moreover, using the feedback polynomial of the LFSR (which is also public), the adversary can specify an error-correction code. Therefore, recovering a_n from z_n is a decoding problem. The are many proposals in the literature on what code best specifies the LFSR output among which low-density parity-check (LDPC) codes seem to be one of the

best candidates [147, 139].

To mount a correlation attack on the WSSC, we consider the modified structure of Figure 5.8c. As we emphasized in Section 5.2.2, in spite of the structural similarities between Figure 5.8c and Figure 5.4a, there are no LFSRs used in the design of the WSSC. Hence, there are no feedback polynomials. In fact, recovering the sequence $z(n)$ from $c(n)$ in Figure 5.8c is not a decoding problem. Therefore, the correlation attack is not well defined for the WSSC.

To mount a correlation attack on the WSSC, we consider the modified structure of Figure 5.8c. Access to the sequence $v(n)$ at point A is necessary to perform decoding. However, $v(n)$ is masked by the transformer $\mathcal{H}$ before appearing at the output. We claim that even with access to this sequence, the correlation attack is unsuccessful. To justify this claim, we note that the register in this structure is not an LFSR, i.e., its state does not evolve based on a linear recurrence equation. Therefore, the adversary is facing with a decoding problem for an unknown code that is more difficult than decoding an arbitrary but known linear code. The latter, known to be an NP-hard problem, is the basis for the security of the McEliece public-key cryptosystem [137]. Therefore, the correlation attack even with access to $v(n)$ fails.

The distinguishing attack is similar to the correlation [75]. The main difference is that no decoding algorithm is employed. Hence, it is expected to be faster than the correlation attack. In a distinguishing attack, the adversary tries to decide whether the data originates from the considered cipher or a random source. To make this decision, hypothesis testing techniques are used. The probability of correct decision is based on the length of observed ciphertext stream. To recover the initial state of the LFSR in a stream cipher, the attacker has to find a low degree multiple of the register polynomial. This attack is irrelevant to the WSSC since the concept of the register polynomial is meaningless in the design.

5.4 Performance Evaluation

In this section, we study the implementation complexity of the proposed WSSC in comparison with the block cipher AES used in the one-bit CFB mode. Results are provided in Table 5.3 for $\kappa = 16$ and $L = 33$. In this table, the measurements for AES are taken from [165]. For a fair comparison, we have assumed that the finite field multiplication and the extended inversion in the design of the WSSC are implemented as table lookups. The third row of the table provides the number of table lookups and the size of the table. For example, $136/(256 \times 256)$ implies 136 lookups in a table of size 256 bits by 256 bits. A similar notation is used in other rows of the table. As the last two rows of this table show, the WSSC has less circuit complexity than the AES. Although the actual performance of a cryptosystem depends on the circuit implementation of its algorithm, the circuit complexity provides some insight about the performance. Based on this justification and Table 5.3, we claim that the WSSC is relatively efficient in both hardware and software. The only drawback is the total amount of memory that is about 4.5 times more than that in AES.

Table 5.3: Complexity comparison between the WSSC and AES in the one-bit CFB mode.

	WSSC	AES
Rounds	2	12
Key size (bits)	128	128
Table lookups / Table size (bits × bits)	$136/(256 \times 256)$, $2/(255 \times 1)$	$160/(8 \times 32)$
XOR, ADD (bits)	134(8)	11(128), 120(32)
Total table lookups (byte)	138	160
Total logical operations (byte)	134	656
Total memory (bytes)	136	16
Table lookups per bit	17.25	20
Logical OPs per bit	16.75	82

5.5 Summary

A new self-synchronizing stream cipher, called WSSC, is proposed in this chapter. In contrast to the traditional designs in which LFSRs are the main ingredients, we employ discrete-time wavelet transforms over fields of characteristic two specifically $\mathsf{GF}(256)$. Lifting to fields higher than the binary field enables the usage of some algebraic tools unavailable over the binary field. The new cipher is iterative, the feature mainly used in block ciphers. There are two discrete-time wavelet transforms used in the structure of a single round of the WSSC. The secret key (typically 128 bits) is used to construct the wavelet transforms. Since the wavelet transform by itself is linear, we include some nonlinear components in our design. We have studied the resistance of the WSSC to many known and relevant attacks (interpolation, algebraic, time-memory tradeoff, divide and conquer, differential, and correlation) and the newly developed delta chosen-ciphertext attack. Our results indicate that one round of the WSSC is vulnerable to the delta attack. To prevent it, we recommend using at least two rounds of the system. We have also evaluated our stream cipher in terms of the performance. Our performance comparisons with the block cipher AES in the one-bit cipher feedback mode reveal that WSSC has lower number of table lookups and logical operations. The cost for improving the circuit complexity is additional memory. As the future work, we plan to implement the WSSC in both software and hardware. We hope that our new approach to the design of self-synchronizing stream ciphers leads to new structures with improved efficient and security.

CHAPTER 6

Paraunitary Public-Key Cryptography

6.1 Background Review

Fell and Diffie [87] were among the pioneers in designing public key cryptosystems based on the difficulty of solving systems of multivariate polynomial equations (Problem 4.4.1). Their first idea was to compose several invertible multivariate mappings each of a special format that we call tame automorphism. A tame automorphism (will be formally defined later in this chapter) is a multivariate mapping over some vector space $\mathbb{F}^n$, where $\mathbb{F}$ is a field and $n \in \mathbb{N}$ is arbitrary, of the form

$$\begin{array}{rccl} \mathbf{t}_i & : & \mathbb{F}^n & \longrightarrow & \mathbb{F}^n \\ & & \mathbf{x} = (x_1, \ldots, x_n) & \longmapsto & \mathbf{y} = (x_1, \ldots, x_{i-1}, x_i + g_i, x_{i+1}, \ldots, x_n) \ . \end{array} \tag{6.1}$$

Here, $g_i \in \mathbb{F}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$ is an arbitrary polynomial in $n-1$ variables and $i \in [\, n \,]$. Given $\mathbf{y} = (y_1, \ldots, y_n) = \mathbf{t}_i(\mathbf{x})$, the inverse is

$$x_j = \begin{cases} y_j, & j \in [\, n \,] \setminus \{\, i \,\} \\ y_i - g_i(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n), & j = i \ . \end{cases} \tag{6.2}$$

Let $\mathbf{t}_{i_1}, \ldots, \mathbf{t}_{i_N}$ be tame automorphisms, where $i_1, \ldots, i_N \in [\, N \,]$ are distinct integers. The OWF proposed in [87] is the multivariate matrix polynomial $\mathbf{t} = \mathbf{t}_{i_1} \circ \cdots \circ \mathbf{t}_{i_N}$. The public information consists of the entries of the matrix polynomial $\mathbf{t}$. The indices $i_1, \ldots, i_N$ and the polynomials $g_{i_1}, \ldots, g_{i_N}$ comprise the trapdoor information. The main problem with this approach is the size of public key that exponentially grows by N. There have been suggestions in [87] to reduce the size of public key such as constructing the OWF over a nilpotent ring or a J ring. Unfortunately, none of these proposals gives a practical system.

6.1.1 Signature Based on Birational Permutations

Consider a multivariate mapping $(x_1, \ldots, x_n) =: \mathbf{x} \mapsto \mathbf{y} := (y_1, \ldots, y_n)$ over a ring $\mathcal{R}$ in which y_i is a low-degree polynomial or rational function of $\mathbf{x}$ for all $i \in [\, n\,]$. This mapping is called birational if it is invertible and its inverse also consists of polynomials or rational functions. Using birational bijections (permutations), a few signature schemes[1] are proposed in [175] over the ring $\mathcal{R} = \mathbb{Z}_k$, where $k = pq$ and p and q are large primes. The idea underlying all the proposed schemes is as follows. Let $\mathbf{f} = (f_1, \ldots, f_n) : \mathbb{Z}_k^n \to \mathbb{Z}_k^n$ be a birational permutation. Ignore the first $s > 0$ polynomials and publish $f_{s+1}, \ldots, f_n$ as public information. To sign a message m, the signer randomly chooses $y_1, \ldots, y_s \in \mathbb{Z}_k$ and sets $y_i = h(m, i)$ for all $i = s+1, \ldots, n$, where h is a publicly known cryptographic hash function. Using her knowledge of deleted polynomials, the signer calculates the signature $\mathbf{x} = \mathbf{f}^{-1}(\mathbf{y})$. This signature is verified if $f_i(\mathbf{x}) = h(m, i)$ for all $i = s+1, \ldots, n$.

The main issue in using this scheme is designing birational permutations. One of the designs suggested in [175] is the triangular birational permutation $\mathbf{f} = (f_1, \ldots, f_n) : \mathbb{Z}_k^n \to \mathbb{Z}_k^n$ defined as

$$\begin{aligned} f_1(\mathbf{x}) &= a_1\, x_1 + g_1 \\ f_2(\mathbf{x}) &= a_2\, x_2 + g_2(x_1) \\ &\vdots \\ f_i(\mathbf{x}) &= a_i\, x_i + g_i(x_1, \ldots, x_{i-1}) \\ &\vdots \\ f_n(\mathbf{x}) &= a_n\, x_n + g_n(x_1, \ldots, x_{n-1}) \ , \end{aligned} \tag{6.3}$$

where $a_i \in \mathbb{Z}_k^\times$ and $g_i \in \mathbb{Z}_k[x_1, \ldots, x_{i-1}]$ for all $i \in [\, n\,]$. To invert $\mathbf{y} = \mathbf{f}(\mathbf{x})$, one starts from y_1 and proceeds to y_n as follows

$$x_i = a_i^{-1}\Big[y_i - g_i(x_1, \ldots, x_{i-1})\Big] \qquad \forall i \in [\, n\,] \ . \tag{6.4}$$

To hide the triangular structure of $\mathbf{f}$, two invertible matrices $\mathbf{A}, \mathbf{B} \in \mathsf{GL}_n(\mathbb{Z}_k)$ are employed.

[1] A formal definition of digital signature is provided in Section 6.7.

The final birational permutation is the mapping $\mathbf{A}\,\mathbf{f}(\mathbf{B}\,\mathbf{x})$. Unfortunately, this scheme proposed in [175] was broken in [38, 39].

6.1.2 Tame Transformation Methods

A public-key cryptosystem based on the composition of four tame automorphisms, called TTM, is introduced by Moh [143, 144]. Let $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$, and $\mathbf{t}_4$ be tame automorphisms and $\mathbf{t}_4 \circ \mathbf{t}_3 \circ \mathbf{t}_2 \circ \mathbf{t}_1 =: \mathbf{f} : \mathbb{F}^n \to \mathbb{F}^n$ be their composition, where $\mathbb{F}$ is a field of characteristic two. The public key consists of the n polynomials $f_1(x_1, \ldots, x_r, 0, \ldots, 0), \ldots, f_n(x_1, \ldots, x_r, 0, \ldots, 0)$ each in r variables for some fixed $r < n$. A signature scheme based on this idea, called TTS, is also proposed in [144, 34]. In the signature scheme, the reduced map $\hat{\mathbf{f}} := (f_1, \ldots, f_r)$ is employed.

The TTM and TTS schemes are broken in [98, 204] in which the cryptanalysis is reduced to an instance of the MinRank problem that can be solved feasibly. The MinRank problem is stated as follows. Given a set of matrices $\mathbf{M}_1, \ldots, \mathbf{M}_t \in \mathsf{M}_n(\mathbb{F})$, find a linear combination $\alpha_1\,\mathbf{M}_1 + \cdots + \alpha_t\,\mathbf{M}_t$, where $\alpha_1, \ldots, \alpha_t \in \mathbb{F}$, that has the minimum rank among all such linear combinations.

6.1.3 Tractable Rational Map Cryptosystem

A public-key cryptosystem based on tractable rational maps, called tractable rational-map cryptosystem (TRMC), is proposed in [195]. The TRMC employs a triangular structure such as the one used by the signature scheme proposed in [175]. A tractable rational map

is a one-to-one affine transformation of the following form

$$\begin{aligned} y_1 &= r_1(x_1) \\ y_2 &= r_2(x_2) \cdot \frac{p_2(x_1)}{q_2(x_1)} + \frac{f_2(x_1)}{g_2(x_1)} \\ &\vdots \\ y_i &= r_i(x_i) \cdot \frac{p_i(x_1, \ldots, x_{i-1})}{q_i(x_1, \ldots, x_{i-1})} + \frac{f_i(x_1, \ldots, x_{i-1})}{g_i(x_1, \ldots, x_{i-1})} \\ &\vdots \\ y_n &= r_n(x_n) \cdot \frac{p_n(x_1, \ldots, x_{n-1})}{q_n(x_1, \ldots, x_{n-1})} + \frac{f_n(x_1, \ldots, x_{n-1})}{g_n(x_1, \ldots, x_{n-1})} \ , \end{aligned} \tag{6.5}$$

where $p_i, q_i, f_i, g_i \in \mathbb{F}[x_1, \ldots, x_{i-1}]$ are polynomials and $r_i[x_i] \in \mathbb{K}[x_i]$ is a rational permutation for all $i \in [\, n\,]$. In (6.5), $\mathbb{F}$ is a finite field and $\mathbb{K}$ is an $\mathbb{F}$-algebra (e.g., an extension field of $\mathbb{F}$ or a matrix algebra). Although attractive, the security of the TRMC has not been well studied.

6.1.4 C^* Algorithm and its Variants

The main challenge in designing multivariate cryptosystems is including a trapdoor in the public polynomials without employing polynomials with special structures. A practical example is the Matsumoto-Imai scheme, called C^*, that uses a quadratic univariate monomial over an extension field of a small finite field [132]. The representation of this monomial over the small field gives a set of quadratic polynomials. Unfortunately, this scheme and many of its variants have been broken because of unexpected algebraic relations and the special structure of the public polynomials. In the following, we briefly describe the main idea of the C^* scheme based on the description provided in [118].

Let $\mathbb{K}$ be an extension field of degree n of the finite field $\mathbb{F}_q$ with characteristic two. Assume $\beta_1, \ldots, \beta_n \in \mathbb{K}$ is a secret basis of $\mathbb{K}$ as an $\mathbb{F}_q$-vector space. Every element $x \in \mathbb{K}$ can be regarded as an n-tuple $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$, where $x = \sum_{i=1}^{n} x_i \, \beta_i$. In transforming the plaintext into the ciphertext, Alice chooses an exponent h of the form

$$h = q^{\theta} + 1 \ , \tag{6.6}$$

where $\theta \in \mathbb{N}$ is a secret exponent, $0 < h < q^n$, and $\mathsf{gcd}(h, q^n - 1) = 1$ [2]. The gcd condition guarantees that the following map is bijective.

$$\begin{array}{rcccl} \phi & : & \mathbb{K} & \longrightarrow & \mathbb{K} \\ & & u & \longmapsto & u^h = u^{q^\theta} u \end{array} \tag{6.7}$$

The inverse map is $\phi^{-1}(v) = v^{h'}$, where $h' = h^{-1} \mod (q^n - 1)$. In addition to ϕ, Alice chooses two secret affine transformations $s, t : \mathbb{F}_q^n \to \mathbb{F}_q^n$, which are bijections over $\mathbb{K}$. The purpose of these affine transformations is to hide the monomial map ϕ; hence, the name "hidden monomial cryptosystem."

The public polynomials of the encryption transformation are obtained by representing the map $\Psi : \mathbb{K} \to \mathbb{K}$, defined as

$$\Psi := t^{-1} \circ \phi \circ s \ , \tag{6.8}$$

as n multivariate polynomials over $\mathbb{F}_q$. To find this representation, we note that the map $u \mapsto u^{q^\theta}$ is an $\mathbb{F}_q$-linear transformation. Thus, there exist a matrix $\mathbf{C} = [\, c_{ij} \,] \in \mathsf{M}_n(\mathbb{F}_q)$ such that $u^{q^\theta} = \sum_{i=1}^n u_i \, \beta_i'$, where

$$\beta_i' = \sum_{j=1}^n c_{ji} \, \beta_j \qquad \forall i \in [\, n \,] \ . \tag{6.9}$$

In addition, it is possible to write the product of two basis elements in terms of the basis, i.e.,

$$\beta_i \, \beta_j = \sum_{k=1}^n m_{ijk} \, \beta_k, \qquad m_{ijk} \in \mathbb{F}_q \tag{6.10}$$

for all $i, j \in [\, n \,]$. Assuming $v = \phi(u)$ and using (6.9) and (6.10), equation (6.7) can be expanded as

$$\sum_{\ell=1}^n v_\ell \beta_\ell = \left(\sum_{i=1}^n u_i \beta_i' \right) \left(\sum_{j=1}^n u_j \beta_j \right) \tag{6.11}$$

that gives

$$v_\ell = \sum_{i,j,k=1}^n c_{ki} \, m_{jk\ell} \, u_i \, u_j \qquad \forall \ell \in [\, n \,] \ . \tag{6.12}$$

[2] If the characteristic of the field were odd, then $\mathsf{gcd}(h, q^n - 1) \geq 2$.

After applying the linear transformations s and t, we get n multivariate polynomials each consisting of a degree-two homogenous part, a linear part, and a constant term.

To decrypt the ciphertext block $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$, Alice, first, calculates $v = t(y) \in \mathbb{K}$, where y is the element in $\mathbb{K}$ corresponding to the vector $\mathbf{y}$. Then, she obtains $u = v^{h'} \in \mathbb{K}$. The plaintext block is the vector representation of $s^{-1}(u)$.

This scheme was broken in [153] using the observation that if we raise both sides of the equation $v = u^h = u^{q^\theta} u$ to $q^\theta - 1$ and multiply both sides by $u\, v$, we get

$$u\, v^{q^\theta} = u^{q^\theta} v \tag{6.13}$$

that leads to equations in $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ that are linear in both sets of variables. Using linear algebra, the cryptanalyst can find these equations without requiring the knowledge of Alice's secret parameters. Solving these equations probably do not uniquely determines the plaintext from the ciphertext. However, it reduces the search for the plaintext to an affine subspace of $\mathbb{F}_q^n$ that is small enough to allow an exhaustive search.

After breaking the Matsumoto-Imai cryptosystem, it has been evolved in many different ways. Some of these variants are described in the following. Unfortunately, many of them are shown to be insecure.

A Hidden-Field Equations (HFE)

In [155], Patarin proposed a new scheme called HFE or little dragon to repair the C^* algorithm. The main different between the C^* algorithm and the HFE scheme is in the exponent h in (6.6). In the HFE, this exponent is

$$h = q^\theta + q^\vartheta - 1 \ , \tag{6.14}$$

where $\theta, \vartheta \in \mathbb{N}$ are secret integers such that $0 < h < q^n$ and $\mathsf{gcd}(h, q^n - 1) = 1$. Since there are not many possibilities for h, it is regarded as a public integer. Because of the special form of the exponent h in (6.14), the characteristic of the ground field $\mathbb{F}_q$ is no longer necessarily even. Hence, it can be any prime. We note that because of the special form of the exponent h in (6.14), the public polynomials are quadratic and, yet, the attack

explained before for the C^* algorithm is unapplicable on the HFE. The only other difference between the C^* algorithm and the HFE is in the affine mappings s and t that are linear in the HFE. Thus, as $\mathbb{F}_q^n$ transformations, there are matrices $\mathbf{A}, \mathbf{B} \in \mathsf{M}_n(\mathbb{F}_q)$ such that $\mathbf{u} = s(\mathbf{x}) = \mathbf{A}\,\mathbf{x}$ and $\mathbf{v} = t(\mathbf{y}) = \mathbf{B}\,\mathbf{y}$.

An efficient attack on the HFE scheme is proposed in [154]. To briefly explain this attack, let $\mathcal{Y} \subset \mathbb{F}_q^n$ be the vector subspace of all ciphertext vectors. The attack in [154] takes advantage of the existence of a bilinear map

$$\begin{array}{rcccc} \diamond & : & \mathcal{Y} \times \mathcal{Y} & \longrightarrow & \mathcal{Y} \\ & & (\mathbf{y}, \mathbf{y}') & \longmapsto & \mathbf{y}'' = \mathbf{y} \diamond \mathbf{y}' \end{array} \tag{6.15}$$

such that if $\mathbf{v} = \mathbf{B}\,\mathbf{y}$, $\mathbf{v}' = \mathbf{B}\,\mathbf{y}'$, and $\mathbf{v}'' = \mathbf{B}\,\mathbf{y}''$, then $v'' = v\,v'$, where v, v', and v'' are the elements in $\mathbb{K}$ corresponding to the vectors $\mathbf{v}, \mathbf{v}'$, and $\mathbf{v}''$, respectively. Using this bilinear map, it is possible to efficiently find a matrix $\mathbf{C} \in \mathsf{M}_n(\mathbb{F}_q)$ such that $\mathbf{x} = \mathbf{C}\,\mathbf{y}^{(h')}$ where

$$\mathbf{y}^{(h')} := \underbrace{\mathbf{y} \diamond \cdots \diamond \mathbf{y}}_{h' \text{ times}} \tag{6.16}$$

and $h' = h^{-1} \mod (q^n - 1)$. Since h' is public, knowledge of $\mathbf{C}$ enables the adversary to decrypt any ciphertext $\mathbf{y} \in \mathcal{Y}$. The details of this attack can be found in [118].

B Big Dragon

To repair the HFE scheme, the big dragon was proposed by Patarin in [154]. In this scheme, as before, $\mathbb{K}$ is a degree-n extension of the finite field $\mathbb{F}_q$ of characteristic two. Similar to the previous schemes, $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ are the plaintext and ciphertext vectors that are related to two intermediate vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}_q^n$ through two linear maps, respectively. Alice chooses an integer h of the form

$$h = q^{\theta_1} + q^{\theta_2} - q^{\vartheta_1} - q^{\vartheta_2} \tag{6.17}$$

such that $\mathsf{gcd}(h, q^n - 1) = 1$. Let $u, v \in \mathbb{K}$ be the elements corresponding to the vectors $\mathbf{u}$ and $\mathbf{v}$, respectively. The relationship between the two intermediate elements u and v is

$$u^{q^{\theta_1} + q^{\theta_2}}\,\phi_1(v) = u^{q^{\vartheta_1} + q^{\vartheta_2}}\,\phi_2(v) \ , \tag{6.18}$$

where $\phi_1, \phi_2 : \mathbb{K} \rightarrow \mathbb{K}$ are affine transformations such that the map $v \mapsto \phi_1(v)/\phi_2(v)$ is bijective. The representation of (6.18) in $\mathbb{F}_q$ gives n quadratic equations in $2n$ variables each of the form

$$\sum_{i,j,k=1}^{n} \gamma_{ijk}\, x_i\, x_j\, y_k + \sum_{i,j=1}^{n} (\xi_{ij}\, x_i\, x_j + \rho_{ij}\, x_i\, y_j) + \sum_{i=1}^{n} (\chi_i\, x_i + \omega_i\, y_i) + \delta = 0 \ , \qquad (6.19)$$

where all the coefficients are public. Since the public polynomials are linear in the y_i variables, they can be efficiently solved by fixing the x_i variables. However, given the ciphertext vector $\mathbf{y} = (y_1, \ldots, y_n)$, the adversary is confronted with a set of quadratic equations. In the other hand, Alice can use her knowledge of the affine transformations ϕ_1 and ϕ_2 to decrypt the ciphertext v as

$$u = \left[\frac{\phi_1(v)}{\phi_2(v)}\right]^{h'} \ , \qquad (6.20)$$

where $h' = h^{-1} \mod (q^n - 1)$. Unfortunately, as explained in [154, 118], the big dragon is often vulnerable to the same type of attacks as the little dragon.

C Efficient Algorithms for Solving Overdefined Systems

As a general cryptanalysis method for solving all systems of quadratic equations, Kipnis and Shamir proposed in [116] the re-linearization technique that efficiently solves a system of public polynomials for the unknowns $x_1, \ldots, x_n$. As explained before, the public polynomials in the C^* algorithm and all its variants consist of n quadratic polynomials in the n variables $x_1, \ldots, x_n$. The proposed cryptanalytic technique, takes advantage of the special structure of quadratic polynomials by replacing every product $x_i\, x_j$ with the new variable w_{ij} for all $1 \leq i < j \leq n$. The result is a large system of linear equations. Since the number of unknowns in the new system is much more than the number of equations, new equations must be generated to solve the linear system using the Gaussian elimination method. New equations are generated using the observation

$$w_{ij}\, w_{k\ell} = w_{ik}\, w_{j\ell} = w_{i\ell}\, w_{jk} \qquad (6.21)$$

for all distinct $i, j, k, \ell \in [\, n\,]$. However, numerical examples showed that most of the newly generated equations are linearly dependent [44]. Hence, the re-linearization algorithm is

less efficient than one may expect. As an improvement to this algorithm, the XL algorithm is proposed in [44] that has polynomial running-time $n^{O(1/\sqrt{\epsilon})}$ for solving any system of ϵn^2 quadratic equations in n variables for all $0 < \epsilon \leq 1/2$. It is also conjectured that the XL algorithm solves any randomly generated system of polynomial equations in sub-exponential time when the number of equations slightly exceeds the number of variables [44]. The difference between the re-linearization and the XL algorithms is that the later generates a set of equations from the given quadratic equations that is larger than the set which the former generates. From this large set, the XL algorithm extracts independent equations, and then applies the linearization algorithm. Hence, the XL algorithm includes the re-linearization as an special case.

A variant of the XL algorithm, called FXL, is introduced in [44]. In this algorithm, some variables are guessed to make the system slightly overdefined. Then, the XL algorithm is applied. The main question is how many variables must be guessed. Although more guesses make the system more unbalanced, they add to the complexity of the algorithm. The optimum number of guesses is provided in [44].

D C^* Family as Signature Schemes

The C^* algorithm and all its successors can be used as signature schemes as well as public-key cryptosystems. Using the same notation as before, we note that the public polynomials in these schemes are derived from the following mapping

$$\phi(t(y)) = s(x)^h \ , \tag{6.22}$$

where s and t are affine or linear transformations and ϕ is the identity map in the C^* and HFE algorithms and $\phi : u \mapsto \phi_1(u)/\phi_2(u)$ in the big dragon. In light of (6.22), the signature corresponding to the plaintext $y \in \mathbb{K}$ is the vector representation of the element $s^{-1}\left(\phi(t(y))^{h'}\right) \in \mathbb{K}$, where $h' = h \mod (q^n - 1)$. A signature $x \in \mathbb{K}$ of the plaintext $x \in \mathbb{K}$ is verified if it satisfies (6.22). Examples of practical signature schemes are QUARTZ [157], FLASH [156], and SFLASH that is a modified version of FLASH [131]. The security of these schemes are studied in [93, 42, 43].

E Some Other Variations

Any algorithm $\mathcal{A}$ that belongs to the set of algorithms introduced before has many variations [155, 158]. The first two simple variations are the $\mathcal{A}-$ and $\mathcal{A}+$ in which k public polynomials are omitted or k randomly generated polynomials are added, respectively. These variations aim to destroy the unexpected algebraic relations, which are employed in many attacks, or blocks attacks such as re-linearization and XL that solve the system of public polynomials. To decrypt a shrunk ciphertext in the $\mathcal{A}-$ algorithm, one performs an exhaustive search on all q^k possible ciphertexts. Hence, the value of k must be small to allow efficient exhaustive search. When $\mathcal{A}-$ is employed is a signature algorithm, k may be large, but not too large to weaken the collision resistance property of the one-way function inherent in algorithm $\mathcal{A}$. The value of k could be very large in the $\mathcal{A}+$ when it is used as an encryption scheme. However, it should not be too large to allow mounting an attack such as re-linearization or XL. When used as a signature scheme, the value of k in the $\mathcal{A}+$ must be small since for a given plaintext, the probability of satisfying the extra equations is q^{-k}. Two other variations are $\mathcal{A}\text{v}-$ and $\mathcal{A}\text{v}+$. In the former, k variables are omitted from the public polynomials while in the later, k variables are added to them. Similar to the previous two variations, the value of k depends on whether the scheme is used as a public-key cryptosystem or as a signature scheme. It is possible to combine the variations in any order to create new variations. For example, $\mathcal{A}-+$ is a variation in which some public polynomials are first omitted and then some randomly generated ones are added. However, none of these variations is guaranteed to increase the security of the original algorithm $\mathcal{A}$. There are cryptanalytic attacks on some of them [158, 41].

6.2 Paraunitary Asymmetric Cryptosystem (PAC)

Throughout this section, we assume

$$\mathbb{L}_n := \mathbb{F}[\mathbf{x}^{\pm 1}] = \mathbb{F}[x_1^{\pm 1}, \ldots, x_n^{\pm 1}] \tag{6.23a}$$

$$\mathbb{P}_n := \mathbb{F}[\mathbf{x}] = \mathbb{F}[x_1, \ldots, x_n] \tag{6.23b}$$

are the ring of Laurent polynomials, defined in Definition 1.1.1, and ordinary polynomials, respectively, in the variable vector $\mathbf{x} = (x_1, \ldots, x_n)$ for some arbitrary $n \in \mathbb{N}$.

The main theme in multivariate public-key cryptography is designing a trapdoor OWF using a set of multivariate polynomials $f_1, \ldots, f_{n'} \in \mathbb{L}_n$ for some $n' \in \mathbb{N}$. The trapdoor information is used to efficiently invert the OWF. To enhance studying these polynomials, we encapsulate them in a single polynomial vector $\mathbf{f} := [\, f_1, \ldots, f_{n'} \,]^\dagger \in \mathbb{L}_n^{n'}$ that can be considered a polynomial mapping $\mathbb{F}^n \to \mathbb{F}^{n'}$ with domain $\mathcal{D}_\mathbf{f}$ and range $\mathcal{R}_\mathbf{f}$. Based on the application, the one-way function $\mathbf{f}$ must have different characteristics as follows.

FACT 6.2.1. *Consider the polynomial mapping $\mathbf{f} : \mathbb{F}^n \to \mathbb{F}^{n'}$ with domain $\mathcal{D}_\mathbf{f}$ and range $\mathcal{R}_\mathbf{f}$.*

(i) If $\mathbf{f}$ is used in a public-key cryptosystem, it must be one-to-one or one-to-many ($n \leq n'$) and $\mathcal{D}_\mathbf{f} = \mathbb{F}^n$.

(ii) If $\mathbf{f}$ is used in a signature scheme, it must be many-to-one ($n \geq n'$) and surjective, i.e., $\mathcal{R}_\mathbf{f} = \mathbb{F}^{n'}$. In addition, for any given $\mathbf{y} \in \mathbb{F}^{n'}$, at least one solution to the equation $\mathbf{y} = \mathbf{f}(\mathbf{x})$ must be efficiently computable.

The challenge in achieving these goals is including a trapdoor in the multivariate polynomials, that contains the inversion information, without using polynomials with special structures that weaken the security of the scheme. In the following, we first introduce a general framework for studying all multivariate polynomial vectors using PU matrices. Based on this framework, we propose two different approaches for designing multivariate trapdoor OWFs suitable for public-key cryptosystems. We show that the first approach results in schemes such as C^* introduced in Section 6.1.4. The second approach leads to an efficient scheme that we call paraunitary asymmetric cryptosystem (PAC) [63].

Consider a set of n randomly-generated polynomials represented by the polynomial vector $\mathbf{f} := [\, f_1, \ldots, f_n \,]^\dagger \in \mathbb{L}_n^n$. Since the column vectors of every polynomial matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{L}_n)$ form an orthonormal basis for the module $\mathbb{L}_n^n$ (Fact 1.2.4), there exists a

polynomial vector $\mathbf{t} \in \mathbb{L}_n^n$ associated with $\mathbf{P}$ such that

$$\mathbf{f}(\mathbf{x}) = \mathbf{P}(\mathbf{x})\,\mathbf{t}(\mathbf{x}) \ . \tag{6.24}$$

In the first approach for designing a trapdoor OWF, we randomly choose a constant unitary matrix $\mathbf{P} \in \mathsf{U}_n(\mathbb{F})$ and set $\mathbf{t}$ equal to a bijection over $\mathbb{F}^n$. In this setting, the mapping $\mathbf{f}$ in (6.24) becomes a bijection over $\mathbb{F}^n$ that satisfies all the conditions requited for both public-key and signature schemes. An instance of schemes designed based on this approach is the C^* algorithm with $\mathbf{t}$ substituted by the polynomial-vector representation of $\phi \circ a$ in (6.8). Similarly, all the successors of C^* such as HFE are representable as in the general formula (6.24). As explained before, most of the HFE-like cryptosystems have been broken.

In the second approach, we choose $\mathbf{P} \in \mathsf{PU}_n(\mathbb{L}_n) \setminus \mathsf{U}_n(\mathbb{F})$ a nonconstant PU polynomial matrix. Similar to the previous approach, the polynomial vector $\mathbf{t}$ is an arbitrary bijection over $\mathbb{F}^n$. The difficulty in using this approach is that for any $\mathbf{y} \in \mathbb{F}^n$, solving the equation $\mathbf{y} = \mathbf{P}(\mathbf{x})\,\mathbf{t}(\mathbf{x})$ for $\mathbf{x}$ requires knowledge of the value of $\mathbf{P}^\dagger(\mathbf{x})$ at $\mathbf{x}$ that in turn requires the knowledge of $\mathbf{x}$. In order to overcome this paradigm, we use an r-variate PU matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{L}_r)$ for some $r \in \mathbb{N}$ with the restriction $1 \leq r \leq n$. This PU matrix is composed with a polynomial vector $\boldsymbol{\varphi}(\mathbf{x}) \in (\mathbb{F}[\mathbf{x}^{\pm 1}])^r$. To decrypt the ciphertext, only the value of $\boldsymbol{\varphi}(\mathbf{x})$ is required. By the definition of PU matrices in Definition 1.2.5, $\mathbf{P}(\mathbf{z})$ is singular at every $\mathbf{z} = (z_1, \ldots, z_r)$ in which $z_i = 0$ for some $i \in [\,r\,]$. This implies that none of the entries of the vector polynomial $\boldsymbol{\varphi}(\mathbf{x})$ must have a root in $\mathbb{F}^n$. The polynomial $\boldsymbol{\varphi}$ is appended to the vector polynomial

$$\boldsymbol{\Psi}_1(\mathbf{x}) := (\mathbf{P} \circ \boldsymbol{\varphi})\,(\mathbf{x})\,\mathbf{t}(\mathbf{x}) \tag{6.25}$$

to form the new vector polynomial

$$\boldsymbol{\Psi}_2(\mathbf{x}) := \begin{bmatrix} (\mathbf{P} \circ \boldsymbol{\varphi})\,(\mathbf{x})\,\mathbf{t}(\mathbf{x}) \\ \boldsymbol{\varphi}(\mathbf{x}) \end{bmatrix} \ . \tag{6.26}$$

To mix the equations, we use the secret affine transformation $\nu(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{A} \in \mathsf{U}_{n+r}(\mathbb{F})$ is a unitary matrix and $\mathbf{b} \in \mathbb{F}^{n+r}$ is an arbitrary vector. A unitary matrix is used since:

1. It can be easily and efficiently generated using Theorem 2.2.1.

2. It is guaranteed to be invertible.

3. Its inverse can be easily obtained with no computation.

In a single formula, the PU trapdoor OWF $\boldsymbol{\Psi}$ is as follows.

$$\begin{array}{rcccl} \boldsymbol{\Psi} & : & \mathbb{F}^n & \longrightarrow & \mathbb{F}^{n+r} \\ & & \mathbf{x} & \longmapsto & \mathbf{y} = \mathbf{A} \begin{bmatrix} (\mathbf{P} \circ \boldsymbol{\varphi})\,(\mathbf{x})\,\mathbf{t}(\mathbf{x}) \\ \boldsymbol{\varphi}(\mathbf{x}) \end{bmatrix} + \mathbf{b} \end{array} \tag{6.27}$$

In the following lemma, we prove that the proposed mapping satisfies all the properties listed in Fact 6.2.1-(i).

LEMMA 6.2.1. *Consider the mapping $\boldsymbol{\Psi}$ in (6.27) with the following specifications:*

(i) $\mathbf{P} \in \mathsf{PU}_r(\mathbb{L}_r)$,

(ii) $\mathbf{t} \in \mathbb{L}_n^n$ *is a bijection over* $\mathbb{F}^n$ *that can be efficiently inverted, and*

(iii) $\boldsymbol{\varphi} \in \left(\mathbb{F}[\mathbf{x}^{\pm 1}]\right)^r$ *such that none of the coordinates of* $\boldsymbol{\varphi}(\mathbf{x})$ *is zero for all* $\mathbf{x} \in \mathbb{F}^n$.

Then, the mapping $\boldsymbol{\Psi}$ is one-to-one with $\mathcal{D}_{\mathbf{f}} = \mathbb{F}^n$ that can be efficiently inverted.

Proof. It is clear that $\mathcal{D}_{\mathbf{f}} = \mathbb{F}^n$. To show that $\boldsymbol{\Psi}$ is one-to-one, let $\mathbf{y} = \boldsymbol{\Psi}(\mathbf{x})$ for some $\mathbf{y} \in \mathcal{R}_{\boldsymbol{\Psi}} \subset \mathbb{F}^{n+r}$. Let $\hat{\mathbf{v}} = \mathbf{A}^\dagger\,(\mathbf{y} - \mathbf{b})$. Set $\hat{\mathbf{v}} = \left[\mathbf{v}^\dagger, \mathbf{z}^\dagger\right]^\dagger$, where $\mathbf{v} = [\,v_1, \ldots, v_n\,]^\dagger$ and $\mathbf{z} = [\,z_1, \ldots, z_r\,]^\dagger$. It can be easily seen that $\mathbf{P}(z)\,\mathbf{t}(\mathbf{x}) = \mathbf{v}$ from which we obtain $\mathbf{t}^{-1}(\mathbf{P}^\dagger(z)\,\mathbf{v})$. Since $\mathbf{x}$ is uniquely obtained from $\mathbf{y}$, the map $\boldsymbol{\Psi}$ in (6.27) is one-to-one. ∎

By this lemma, the OWF $\boldsymbol{\Psi}$ can be used a public-key cryptosystem. The trapdoor information consist of the PU matrix $\mathbf{P}$, the unitary matrix $\mathbf{A}$, the vector $\mathbf{b}$, the automorphism $\mathbf{t}$, and the multivariate polynomial $\boldsymbol{\varphi}$. Algorithm 6.1 and Algorithm 6.2 are used to encrypt and decrypt in the PAC cryptosystem.

The proposed scheme operates on any finite field $\mathsf{GF}(2^m)$ with $m \geq 2$. The reason it should not be used over $\mathsf{GF}(2)$ is that since none of the entries of the vector polynomial $\boldsymbol{\varphi}$

Algorithm 6.1: PAC encryption.

Input: Plaintext block $\mathbf{x} \in \mathbb{F}^n$

Output: Ciphertext block $\mathbf{y} \in \mathbb{F}^{n+r}$

1. Evaluate the public vector-polynomial $\boldsymbol{\Psi}(\mathbf{x})$ at $\mathbf{x}$.

Algorithm 6.2: PAC decryption.

Input: Ciphertext block $\mathbf{y} \in \mathbb{F}^{n+r}$

Output: Plaintext block $\mathbf{x} \in \mathbb{F}^n$

1. $\hat{\mathbf{v}} \leftarrow \mathbf{A}^{\dagger}\,(\mathbf{y} + \mathbf{b}) \in \mathbb{F}^{n+r}$
2. $\mathbf{v} \leftarrow [\,v_1, \ldots, v_n\,]^{\dagger}$ and $\mathbf{z} \leftarrow [\,z_1, \ldots, z_r\,]^{\dagger}$, where $\hat{\mathbf{v}} = \left[\,\mathbf{v}^{\dagger}, \mathbf{z}^{\dagger}\,\right]^{\dagger}$
3. $\mathbf{x} \leftarrow \mathbf{t}^{-1}\left(\mathbf{P}^{\dagger}(\mathbf{z})\mathbf{v}\right)$

must take the value zero, the only possible choice is $\boldsymbol{\varphi}(\mathbf{x}) \equiv \mathbb{1}_{n,1}$. With this choice, the PU matrix $\mathbf{P}$ becomes a constant matrix independent of $\mathbf{x}$. In this setting, the second approach reduces to the firs one that seems to be insecure. We suggest using $\mathsf{GF}(256)$ to enhance the implementation of the scheme.

In the rest of this section, we study efficient and practical methods to generate different components of the PAC and also the setup algorithms.

6.2.1 Bijective Mappings

In this section, we address the issue of generating bijective polynomial-vectors in $\mathbb{P}_n^n$ required in the design of the PAC. It can be easily verified that the $\mathbb{P}_n$-module of endomorphisms over $\mathbb{P}_n$ is isomorphic to the module $\mathbb{P}_n^n$ [187]. The immediate consequence of this result is that every bijection over $\mathbb{F}^n$ is uniquely representable by an isomorphism over $\mathbb{P}_n$. Hence, we can reduce the study of all polynomial vectors in $\mathbb{P}_n^n$ that are bijective over $\mathbb{F}^n$, denoted by $\mathsf{B}(\mathbb{F}^n)$, to the $\mathbb{P}_n$-module of all automorphisms over $\mathbb{P}_n$ denoted by $\mathsf{Aut}(\mathbb{P}_n)$. Two important subgroups of $\mathsf{Aut}(\mathbb{P}_n)$ are:

(*i*) the group of all affine automorphisms

$$\mathsf{Aff}(\mathbb{P}_n^n) := \left\{ \mathbf{A}\mathbf{x} + \mathbf{b} \,:\, \forall \mathbf{A} \in \mathsf{GL}_n(\mathbb{F}), \mathbf{b} \in \mathbb{F}^n \right\} \tag{6.28}$$

and

(*ii*) the group of all *elementary automorphisms* $\mathsf{EA}(\mathbb{P}_n^n)$. An elementary automorphism is a polynomial vector of the form

$$\mathbf{e}(\mathbf{x}) := [\, x_1, \ldots, x_{i-1}, x_i + g, x_{i+1}, \ldots, x_n \,]^\dagger \quad , \tag{6.29}$$

where $i \in [\, n \,]$ is an arbitrary index and $g \in \mathbb{F}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$. (We note that g is a multivariate polynomial that is independent of x_i.)

All finite compositions of affine and elementary automorphisms generate a group that has practical importance [187].

Definition 6.2.1 (Tame Automorphism). *The subgroup of* $\mathsf{Aut}(\mathbb{P}_n)$ *generated by all finite compositions of affine and elementary automorphisms is called the group of* tame automorphisms *denote by*

$$\mathsf{TA}(\mathbb{P}_n^n) := \left\{ \mathbf{f}_1 \circ \cdots \circ \mathbf{f}_\ell \,:\, \mathbf{f}_i \in \mathsf{Aff}(\mathbb{P}_n^n) \cup \mathsf{EA}(\mathbb{P}_n^n) \quad \forall \ell \in \mathbb{N},\ \forall i \in [\, \ell \,] \right\} . \tag{6.30}$$

The practical interest in tame automorphisms comes from the fact that both affine and elementary automorphisms can be efficiently inverted. Hence, tame automorphisms have the same property. An interesting question is whether all automorphisms over $\mathbb{P}_n$ are tame. The following result answers this question [133].

Theorem 6.2.1. *For all fields* $\mathbb{F}$ *of characteristic two excluding the binary field* $\mathbb{F}_2$ *and for all* $n \geq 2$, $|\mathsf{TA}(\mathbb{P}_n^n)| = |\mathsf{B}(\mathbb{F}^n)|\,/2$. *In fact,* $\mathsf{TA}(\mathbb{P}_n^n)$ *is the alternating subgroup*[3] *of* $\mathfrak{S}_{|\mathbb{F}^n|}$.

This result shows that by using only tame automorphisms, we are missing out half of all bijections.

[3]The alternating group consists of all even permutations.

As a conclusion, we suggest using the following tame automorphism for the bijection $\mathbf{t} = [\, t_1, \ldots, t_n \,]^{\dagger}$ in the design of the PAC.

$$t_i(\mathbf{x}) = \alpha_i \, x_{\sigma(i)} + g_i(x_{\sigma(1)}, \ldots, x_{\sigma(i-1)}) \qquad \forall i \in [\, n \,] \tag{6.31}$$

Here, $\sigma \in \mathfrak{S}_n$ is a public permutation, $\alpha_i \in \mathbb{F}^{\times}$, and $g_i \in \mathbb{F}[x_{\sigma(1)}, \ldots, x_{\sigma(i-1)}]$ for all $i \in [\, n \,]$. To efficiently solve $\mathbf{y} = \mathbf{t}(\mathbf{x})$ for $\mathbf{x}$ for a given $\mathbf{y} \in \mathbb{F}^n$, the following formula is recursively used starting from $i = 1$.

$$x_{\sigma(i)} = \alpha_i^{-1} \Big(y_i + g_i(x_{\sigma(1)}, \ldots, x_{\sigma(i-1)}) \Big) \qquad \forall i \in [\, n \,] \tag{6.32}$$

6.2.2 Polynomial Vector φ

The composite matrix polynomial $(\mathbf{P} \circ \boldsymbol{\varphi})(\mathbf{x})$ in (6.27) approximates the PU matrix $\mathbf{P}(\mathbf{x})$ in (6.24). This approximation is in the sense that the entries of the PU matrix $\mathbf{P}(\mathbf{x})$ in (6.24) are taken from the ring $\mathbb{F}[\mathbf{x}]$ while those of $(\mathbf{P} \circ \boldsymbol{\varphi})(\mathbf{x})$ in (6.27) belong to the ring $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})]$. These two rings are both extensions of the finite field $\mathbb{F}$, and their relationship is expressed by

$$\mathbb{F} \subseteq \mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})] \subseteq \mathbb{F}[\mathbf{x}] \ . \tag{6.33}$$

To determine the extension degree of $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})]$, we use the notion of *transcendence degree* that generalizes the notion of the dimension of a vector space [105, 122].

Definition 6.2.2 (Transcendence Degree). *Let $\mathcal{S}$ be a subset of $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})]$ that is algebraically independent over $\mathbb{F}$ and is maximal with respect to the set-theoretic inclusion among the set of all algebraically independent subsets of $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})]$. The cardinality of $\mathcal{S}$ is an integer between 0 and n that is called the transcendence degree of $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})]$ over $\mathbb{F}$.*

Let d be the transcendence degree of the extension ring $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})]$. If $d = 0$, then $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})] \cong \mathbb{F}$, and if $d = n$, $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})] \cong \mathbb{F}[\mathbf{x}]$. It can be easily verified that if $\boldsymbol{\varphi}(\mathbf{x})$ is *invertible*, then $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})] \cong \mathbb{F}[\mathbf{x}]$, i.e., $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})]$ achieves its highest transcendence degree [187]. A polynomial vector $\mathbf{f} \in \mathbb{L}_n^n$ is called invertible if there exists a polynomial vector $\mathbf{g} \in \mathbb{L}_n^n$ such that $(\mathbf{g} \circ \mathbf{f})(\mathbf{x}) \equiv \mathbf{x}$. We note that both affine and elementary automorphisms

are invertible. Thus, tame automorphisms have the same property. Since in practice usually $r < n$, $\boldsymbol{\varphi}(\mathbf{x})$ cannot be invertible that implies $d < n$. For $\mathbb{F}[\boldsymbol{\varphi}(\mathbf{x})]$ to achieve the highest possible transcendence degree, which is r, we design $\boldsymbol{\varphi}(\mathbf{x})$ to be "semi-invertible", i.e., after evaluating it at $n-r$ coordinates of $\mathbf{x}$, the resulting r-variate polynomial becomes invertible. If $\boldsymbol{\varphi} = [\,\varphi_1, \ldots, \varphi_r\,]^\dagger$, then we suggest using the following composition

$$\varphi_i = \chi \circ \rho_i \qquad \forall i \in [\,r\,] \quad . \tag{6.34}$$

Here, $\boldsymbol{\rho} = [\,\rho_1, \ldots, \rho_r\,]^\dagger \in (\mathbb{F}[\mathbf{x}])^r$ is close to a bijection and $\chi \in \mathbb{F}[x]$ is a polynomial irreducible over $\mathbb{F}$. We may use the vector polynomial

$$\rho_i(\mathbf{x}) = \alpha_i\, x_{\sigma(r-i+1)} + g_i(x_{\sigma(r-i+2)}, \ldots, x_{\sigma(n)}) \qquad \forall i \in [\,r\,] \quad , \tag{6.35}$$

where $\sigma \in \mathfrak{S}_n$, $\alpha_i \in \mathbb{F}^\times$, and $g_i \in \mathbb{F}[x_{\sigma(r-i+2)}, \ldots, x_{\sigma(n)}]$ for all $i \in [\,r\,]$. In order to invert $\boldsymbol{\rho}$ in (6.35), the values of $x_{\sigma(r+1)}, \ldots, x_{\sigma(n)}$ can be arbitrarily chosen and then the values of $x_{\sigma(1)}, \ldots, x_{\sigma(r)}$ are obtained from a recursive equation. Hence, $|\boldsymbol{\rho}^{-1}(\mathbf{z})| = |\mathbb{F}|^{n-r}$ for any $\mathbf{z} \in \mathbb{F}^r$ that implies $\boldsymbol{\rho}$ in (6.34) is "close" to a bijection.

As explained before, none of the entries of the vector polynomial $\boldsymbol{\varphi}$ must have a root in $\mathbb{F}^n$. The irreducible polynomial $\chi \in \mathbb{F}[x]$ in (6.34) is used to guarantee that this does not happen. We suggest using the polynomial

$$\chi(x) = x^2 + x + \omega, \qquad \omega \in \mathbb{F} \tag{6.36}$$

that is irreducible whenever $\mathsf{Tr}(\omega) := \sum_{k=0}^{m-1} \omega^{2^k} \neq 0$ assuming $\mathbb{F} = \mathsf{GF}(2^m)$ [136]. Since χ is not an automorphism, $\boldsymbol{\varphi}$ is not as close to a bijection as $\boldsymbol{\rho}$ is. However, χ is a two-to-one mapping since $\chi(x+1) = \chi(x)$ for all $x \in \mathbb{F}$. Hence, the vector polynomial $\boldsymbol{\varphi}$ is not significantly deviated from a bijection.

6.2.3 Setup Algorithms

In this section, we explain the algorithms required to setup the PAC. The *master secret-key* provided by the user is a vector $\mathbf{k}$ of length n that is uniformly at random chosen from

$\mathbb{F}^n$. The bit length of the master secret-key is $n|\mathbb{F}|$. In a typical setting, $n = 16$ and the underlying finite-field is $\mathsf{GF}(256)$. In this situation, the master secret-key is 128 bits long. The secret vectors required to be specified for all the components of the PAC are derived from the master secret-key by using the key-expansion algorithm in Algorithm 5.2. The output of this algorithm is the set $\mathcal{K}$ consisting of T secret vectors. The integer κ is chosen such that there are enough vectors in $\mathcal{K}$ for the specification of all components of PAC.

It is possible to replace the proposed key-expansion algorithm with a pseudo-random number generator. For our purpose, a fast pseudo-random number generator such as the shrinking or self-shrinking generator, explained in Chapter 5, is adequate [37, 135].

Algorithm 6.3 is used to generate the public and secret information in the PAC.

Algorithm 6.3: PAC key generation.

Input: Master secret-key $\mathbf{k} \in \mathbb{F}^n$

Public Output: Polynomial vector $\mathbf{\Psi} \in (\mathbb{F}[\mathbf{x}])^{n+r}$

Secret Output: $\mathbf{P}$: an r-variate PU matrix in $\mathsf{PU}_n(\mathbb{F}[\mathbf{z}])$, $\boldsymbol{\varphi}$: a vector polynomial in $(\mathbb{F}[\mathbf{x}])^r$, $\mathbf{t}$: an automorphism in $\mathsf{Aut}(\mathbb{F}[\mathbf{x}])$, $\mathbf{A}$: a unitary matrix in $\mathsf{U}_{n+r}(\mathbb{F})$, $\mathbf{b}$: a constant vector in $\mathbb{F}^{n+r}$.

1. Use Algorithm 5.2 to generate the set $\mathcal{K}$.
2. Generate an r-variate PU matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{F}[\mathbf{z}])$ by multiplying elementary building blocks whose parameters are taken from the set $\mathcal{K}$.
3. Choose a semi-invertible vector polynomial $\boldsymbol{\varphi} \in (\mathbb{F}[\mathbf{x}])^r$ such that none of its entries, as a polynomial in $\mathbb{F}[\mathbf{x}]$, has a root in $\mathbb{F}^n$. Use the vectors in $\mathcal{K}$ as the design parameters. (The composition in (6.34) might be used.)
4. Choose an automorphism $\mathbf{t} \in \mathsf{Aut}(\mathbb{F}[\mathbf{x}])$ using vectors in $\mathcal{K}$ as its coefficients.
5. Construct the vector polynomials $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ as in (6.25) and (6.26), respectively.
6. Generate a unitary matrix $\mathbf{A} \in \mathsf{U}_{n+r}(\mathbb{F})$ by multiplying the elementary building blocks with different design parameters. In addition, choose a vector $\mathbf{b} \in \mathbb{F}^{n+r}$ using the vectors in $\mathcal{K}$.
7. Construct the vector polynomial $\mathbf{\Psi}(\mathbf{x})$ as in (6.27).

6.3 Probabilistic PAC

The proposed PAC is a deterministic scheme, i.e., the mapping from the plaintext space to the ciphertext space is deterministic. In other words, given the plaintext, the corresponding ciphertext is always the same. This determinism might cause some leakage of partial information to the adversary. For example, the RSA function preserves the Jacobi symbol of the plaintext, and with the discrete-log function, it is easy to compute the least significant bit of the plaintext from the ciphertext by a simple Legendre symbol calculation [95]. To prevent the leakage of partial information, the notion of *semantic security* is proposed in [96].

Informally, a public-key cryptosystem is semantically secure if, for all probability distributions over the message space, whatever a passive adversary can compute in expected polynomial time about the given ciphertext, it can compute in expected polynomial time without the ciphertext [96, 137]. Semantic security is the reminiscent of Shannon's perfect secrecy in which the adversary is given unbounded computational power. Although theoretically attractive, perfect secrecy is not achievable unless the key is as long as the message. This requirement hinders the practical usefulness of perfect secrecy. By contrast, semantic security can be viewed as the polynomially-bounded version of perfect secrecy in which the adversary is given limited computational power.

In a semantically secure cryptosystem, the mapping from the plaintext to the ciphertext is probabilistic. Hence, different encryptions give different ciphertexts corresponding to a single plaintext. An efficient probabilistic public-key cryptostsem based on the RSA one-way function is proposed in [21]. In general, there are standard methods to construct probabilistic schemes based on deterministic one-way functions. In the following, we briefly explain the method proposed in [15] to achieve semantic security. This method is based on the random oracle model.

Let $\mathbf{G} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ be a random generator that is public to everybody and $\mathbf{\Psi}$ be a

TOWF such as the one in (6.27). Consider the following probabilistic encryption function.

$$\begin{array}{rcccl} \mathbf{E}^{\mathbf{G}} & : & \mathbb{F}^n & \longrightarrow & \mathbb{F}^{2n+r} \\ & & \mathbf{x} & \longmapsto & \mathbf{\Psi}(\mathbf{u}) || \left(\mathbf{G}(\mathbf{u}) + \mathbf{x}\right) \quad . \end{array} \tag{6.37}$$

Here, $\mathbf{u} \in \mathbb{F}^n$ is a randomly chosen vector and $||$ denotes the concatenation of two vectors. It is proved in [15] that the encryption function $\mathbf{E}^{\mathbf{G}}$ is semantically secure in the random oracle model. Note that the data expansion factor of $2n + r$ is unavoidable. The decryption algorithm is straightforward: The receiver, who knows the trapdoor, uses his information about $\mathbf{\Psi}^{-1}$ to obtain $\mathbf{u}$, applies the random generator $\mathbf{G}$ to $\mathbf{u}$, and then easily obtains the message $\mathbf{x}$ from $\mathbf{G}(\mathbf{u}) + \mathbf{x}$. The adversary, without the trapdoor information, is unable to calculate $\mathbf{u}$ and hence $\mathbf{x}$ although $\mathbf{G}$ is public.

6.4 On the Computational Security of the PAC

In this section, we study the computational security of the PAC by providing evidences that relate the difficulty of inverting the underlying OWF in the PAC to a known computationally hard problem. The computational security measures the amount of computational effort required, by the best currently-known methods, to defeat a system [137]. In general, it is very difficult to prove the security of public-key cryptosystems [137, 179]. For example, it is known that if the public modulus in RSA is factored into its prime factors, then RSA can be broken. However, it is not proved that breaking RSA is equivalent to factoring the public modulus [26, 95]. By providing some theorems and conjectures, in this section, we establish a connection between the hardness of inverting the mapping $\mathbf{\Psi}$ in the proposed PAC and the difficulty of solving a general system of multivariate polynomial equations. We note that the general framework for designing multivariate cryptosystems based on PU matrices, presented in the previous section, has made the following mathematical analysis possible. We would also like to emphasize that this section only focuses on the one-way property of the mapping $\mathbf{\Psi}$. The concept of "semantic security" was addressed in Section 6.3.

As explained in the previous section, the proposed public-key cryptosystem PAC is

based on the general formulation (6.24), where $\mathbf{f} \in \mathbb{P}_n^n$ is an arbitrary polynomial vector, $\mathbf{P} \in \mathsf{PU}_n(\mathbb{P}_n)$, $\mathbf{t} \in \mathsf{Aut}(\mathbb{P}_n)$, and $\mathbb{P}_n = \mathbb{F}[\mathbf{x}]$. Since the column vectors of any PU matrix form an orthonormal basis for the module $\mathbb{P}_n^n$, given an arbitrary polynomial vector $\mathbf{f}$, the relation (6.24) is always valid when the automorphism condition on $\mathbf{t}$ is relaxed. To study the relationship between all polynomial vectors generated as in (6.24) and the module of all polynomial vectors $\mathbb{P}_n^n$, let $\mathsf{PAC}_n(\mathbb{F})$ be the set of all such vectors, i.e.,

$$\mathsf{PAC}_n(\mathbb{F}) := \left\{ \mathbf{f} \in \mathbb{P}_n^n \,:\, \exists\, \mathbf{P} \in \mathsf{PU}_n(\mathbb{P}_n) \text{ and } \mathbf{t} \in \mathsf{Aut}(\mathbb{P}_n) \text{ such that } \mathbf{f} = \mathbf{P}\,\mathbf{t} \right\} . \qquad (6.38)$$

Clearly, $\mathsf{PAC}_n(\mathbb{F}) \subseteq \mathbb{P}_n^n$. The security of our proposed scheme (assuming $r = n$) reduces to the difficulty of solving a randomly-generated system of multivariate polynomial equations if and only if equality holds.

Conjecture 6.4.1. *In the description of the PAC, assume $r = n$. Then, $\mathsf{PAC}_n(\mathbb{F}) = \mathbb{P}_n^n$. In other words, given an arbitrary polynomial vector $\mathbf{f} \in \mathbb{P}_n^n$, there always exists a matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{P}_n)$ and an automorphism $\mathbf{t} \in \mathsf{Aut}(\mathbb{P}_n)$ such that (6.24) holds.*

This conjecture implies that an arbitrary system of multivariate polynomial equations can always be represented as an instance of the proposed PAC. Hence, if proved to be valid, this conjecture shows that the public polynomials in the PAC are *indistinguishable* from an arbitrary system of multivariate polynomial equations.

To investigate Conjecture 6.4.1, we note that the group $\mathsf{PU}_n(\mathbb{P}_n)$ acts on the module $\mathbb{P}_n^n$ by matrix multiplication. By the norm-preservation property of PU matrices in Lemma 1.2.2, the group $\mathsf{PU}_n(\mathbb{P}_n)$, in fact, acts on the set $\mathcal{V}_n^\alpha(\mathbb{P}_n) := \left\{ \mathbf{f} \in \mathbb{P}_n^n \,:\, \mathbf{f}^\dagger \mathbf{f} = \overline{\alpha}\alpha \right\}$ for every $\alpha \in \mathbb{P}_n$. This group action is transitive if for every two arbitrary $\mathbf{f}, \mathbf{t} \in \mathcal{V}_n^\alpha(\mathbb{P}_n)$, there exists a matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{P}_n)$ such that $\mathbf{f} = \mathbf{P}\mathbf{t}$. The statement of Conjecture 6.4.1 is weaker than the transitivity of the action of the group $\mathsf{PU}_n(\mathbb{P}_n)$ on $\mathcal{V}_n^\alpha(\mathbb{P}_n)$ since to prove the indistinguishability of the public polynomials in the PAC, the polynomial vector $\mathbf{t}$ is not required to be a fixed vector, but it suffices to be any arbitrary automorphism. Hence, the proof of the following conjecture, immediately proves Conjecture 6.4.1.

CONJECTURE 6.4.2 (Transitivity of the Action of the PU Group). *The group* $\mathsf{PU}_n(\mathbb{P}_n)$ *acts transitively on the set* $\mathcal{V}_n^{\alpha}(\mathbb{P}_n)$.

This conjecture has strong ties with the *PU completion problem* stated below[4] [152, 122].

PROBLEM 6.4.1 (PU Completion Problem). *Given the vector* $\mathbf{f} \in \mathbb{P}_n^n$ *such that* $\mathbf{f}^{\dagger}\mathbf{f} = \overline{\alpha}\alpha$, *where* $\alpha \in \mathbb{P}_n$, *does there exist a matrix* $\mathbf{P} \in \mathsf{PU}_n(\mathbb{P}_n)$ *such that* $\mathbf{f}$ *is the first column of* $\alpha\mathbf{P}$*?*

The equivalence between the transitivity of the action of $\mathsf{PU}_n(\mathbb{P}_n)$ and the PU completion problem is proved in the following lemma.

LEMMA 6.4.1. *The group* $\mathsf{PU}_n(\mathbb{P}_n)$ *acts transitively on* $\mathcal{V}_n^{\alpha}(\mathbb{P}_n)$ *for every* $\alpha \in \mathbb{P}_n$ *if and only if the PU completion problem has a positive answer.*

Proof. To prove the ($\Rightarrow$) direction, let $\mathbf{f} \in \mathbb{P}_n^n$ such that $\mathbf{f}^{\dagger}\mathbf{f} = \overline{\alpha}\alpha$. Since $\mathsf{PU}_n(\mathbb{P}_n)$ acts transitively on $\mathcal{V}_n^{\alpha}(\mathbb{P}_n)$, there exists a matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{P}_n)$ such that $\mathbf{f} = \mathbf{P}\alpha\mathbf{e}_n^1$, where $\mathbf{e}_n^1$ is defined in Table 1.3. The first column of $\alpha\mathbf{P}$ is $\mathbf{f}$ and $\alpha\mathbf{P}$ is the PU completion of $\mathbf{f}$. For the ($\Leftarrow$) direction, let $\mathbf{f}, \mathbf{g} \in \mathcal{V}_n^{\alpha}(\mathbb{P}_n)$ be arbitrary polynomial vectors. Since every PU polynomial vector has a PU completion, there are PU matrices $\mathbf{P}, \mathbf{Q} \in \mathsf{PU}_n(\mathbb{P}_n)$ such that $\mathbf{f} = \alpha\mathbf{P}\mathbf{e}_n^1$ and $\mathbf{g} = \alpha\mathbf{Q}\mathbf{e}_n^1$. This implies $\mathbf{f} = \mathbf{P}\mathbf{Q}^{\dagger}\mathbf{g}$, so $\mathsf{PU}_n(\mathbb{P}_n)$ acts transitively on $\mathcal{V}_n^{\alpha}(\mathbb{P}_n)$. ∎

The PU completion problem has a positive answer if the class of generalized-paraunitary matrices, denoted by $\mathsf{GPU}_n(\mathbb{P}_n)$, is considered instead of the class of PU matrices [152, 122].

THEOREM 6.4.1 (Quillen-Suslin). *Every generalized-paraunitary polynomial-vector* $\mathbf{f} \in \mathbb{P}_n^n$ *has a completion in* $\mathsf{GPU}_n(\mathbb{P}_n)$.

The matrix $\mathbf{P} \in \mathsf{M}_{n,k}(\mathbb{P}_n)$ is called generalized PU if there exists a matrix $\mathbf{Q} \in \mathsf{M}_{n,k}(\mathbb{P}_n)$ such that $\mathbf{Q}^{\dagger}\mathbf{P} = \mathbf{I}_k$. The set of all $n \times n$ generalized PU matrices with entries in the ring

[4] We emphasize that this is not a computational problem. In other words, the problem is not about finding an algorithm to complete any PU vector into a square PU matrix. The question raised in this problem is only about the possibility of such completion.

$\mathbb{P}_n$ is denoted by $\mathsf{GPU}_n(\mathbb{P}_n)$. We note that $\mathsf{PU}_n(\mathbb{P}_n) \subseteq \mathsf{GPU}_n(\mathbb{P}_n)$. The PU completion problem has a positive answer for the case $n = 2$, but for $n > 2$, it is still an open problem [152]. This problem also has a positive answer for arbitrary n when $\mathbb{P}_n = \mathbb{C}[\mathbf{x}]$ [152].

Collectively, the results obtained so far can be summarized as follows.

FACT 6.4.1. *If the PU completion problem (Problem 6.4.1) is shown to have a positive answer, then the indistinguishability conjecture holds (i.e.,* $\mathsf{PAC}_n(\mathbb{F}) = \mathbb{P}_n^n$*). However, the inverse is not necessarily true.*

The OWF $\boldsymbol{\Psi}$ in (6.27) is slightly different from the general formula (6.24) in two ways:

1. The PU matrix $\mathbf{P}(\mathbf{x})$ is approximated by the composite matrix polynomial $(\mathbf{P} \circ \boldsymbol{\varphi})(\mathbf{x})$.
2. The vector polynomial $\boldsymbol{\varphi}(\mathbf{x})$ is concatenated to the end of the vector polynomial $\boldsymbol{\Psi}(\mathbf{x})$ in (6.25).

This deviation might seem as a source of leaking partial information in the ciphertext. In order to address this issue, we note the following points.

Approximating the PU matrix $\mathbf{P}(\mathbf{x})$ with the composite matrix polynomial $(\mathbf{P} \circ \boldsymbol{\varphi})(\mathbf{x})$ is necessary in order to design an efficient protocol. Security and complexity are often in the same direction: increasing one increases the other one. In order to achieve a practical system, a tradeoff between the security and complexity must be established. This is not only specific to the PAC. For example, the idea in RSA is using the hardness of factoring general composite integers. In theory, the modulus can be any composite integer. However, in practice, only composite integers with only two distinct prime factors are used. Without this simplification, key generation in RSA could be time consuming. The ElGamal digital signature scheme is another example. The underlying hard problem in this scheme is the discrete-log problem. However, the trapdoor OWF employed in this scheme is such that for selective forgery, solving the discrete-log problem is not required. It seems that the general strategy in designing one-way functions consists of the following steps:

1. choose a mathematical problem that is believed to be hard,

2. design a trapdoor OWF based on this hard problem, and

3. make some simplification to the general scheme to design an efficient and practical scheme.

We have followed these guidelines in designing the PAC.

To address the concatenation issue, we note that the coefficients of the elements constructing $\mathbf{\Psi}_1(\mathbf{x})$ are calculated using the expanded key. Hence, they are unavailable to the adversary. Moreover, the unitary matrix $\mathbf{A}$ and the vector $\mathbf{b}$ are also secret. Therefore, there does not seem to exist an easy way for the adversary to obtain the coefficients of the vector polynomial $\boldsymbol{\varphi}(\mathbf{x})$ from the public polynomials.

In the worst case, the adversary has somehow obtained the vector polynomial $\boldsymbol{\varphi}(\mathbf{x})$. Since $\boldsymbol{\varphi}$ is a many-to-one mapping, its value cannot be used to calculate the plaintext $\mathbf{x}$. Another approach for the adversary to decrypt the ciphertext is obtaining the PU matrix $\mathbf{P}(\mathbf{z})$. In this approach, the adversary is facing with the following question: "How to find the coefficients of the PU matrix $\mathbf{P}(\mathbf{z})$ from $\mathbf{\Psi}_1(\mathbf{x})$?" This problem seems to be strongly related to the polynomial decomposition problem which is as follows.

PROBLEM 6.4.2 (Polynomial Decomposition Problem). *Given a vector polynomial* $\mathbf{f} \in R^n$*, determine whether there exist vector polynomials* $\mathbf{g}, \mathbf{h} \in R^n$ *such that* $\mathbf{f} = \mathbf{g} \circ \mathbf{h}$*. In the affirmative case, compute* $\mathbf{g}$ *and* $\mathbf{h}$*.*

This problem has been studied extensively in the literature [66, 89, 90, 120, 192]. There are polynomial-time algorithms for Problem 6.4.2 in the univariate case. In the multivariate case, there are only solutions for cases when either $\mathbf{g}$ or $\mathbf{h}$ is univariate. A general solution to the multivariate case seems to be difficult [192]. The instance of Problem 6.4.2 in which the characteristic of the field divides the degree of the polynomial $\mathbf{g}$ is called *wild case*. There is only a sub-exponential solution to the univariate version of the problem in the wild case [90]. We note that the characteristic of the field in case of the OWF $\mathbf{\Psi}$ is two. If the degree of the PU matrix $\mathbf{P}$ is even, then we have a wild case of Problem 6.4.2. Hence, concatenating $\boldsymbol{\varphi}(\mathbf{x})$ at the end of $\mathbf{\Psi}_1(\mathbf{x})$ does not seem to leak any information.

6.5 A Practical Instance of the PAC

The PAC proposed in Section 6.2 by Algorithm 6.3 is the template of a multivariate public-key cryptosystem since some components of the scheme (such as the automorphism $\mathbf{t}$) are not precisely specified although some suggestions were made. Therefore, there are numerous ways to design a public-key scheme based on the proposed template. A good design should meet the following criteria:

1. Exploiting the structure of the public polynomials in an attack without knowledge of the secret information should be infeasible. In other words, structures in which the public polynomials can be efficiently solved by an adversary must be avoided.
2. To reduce both the size of the memory required to store the public polynomials and the time to encrypt and to decrypt, the public polynomials should be sparse.
3. The automorphism $\mathbf{t}$ should be efficiently invertible.

In this section, we present an instance of the PAC that satisfies the suggested properties. In this instance, we use $\mathbb{F} = \mathsf{GF}(256)$ as the underlying field to enhance the byte-oriented implementation of the scheme. To eliminate the possibility of performing a feasible exhaustive search on the entire key space, we choose $n = 16$. With this choice, the master secret-key consists of 16 symbols from $\mathsf{GF}(256)$ that leads to a key space of size 2^{128}. The proposed scheme is very flexible and these settings can be easily changed. We fix the value of r in our design. Considering the value of n, we suggest $r = 10$ for reasons that will be explained later in this chapter. With these choices, the size of the ciphertext block becomes 208 bits.

6.5.1 Constructing the Polynomial Vector Ψ

For the PU matrix, we use only the degree-one and the degree-two building blocks defined in (2.9) and (2.10) because they have less parameters than the degree-$M\tau$ building block in (2.12). Moreover, the degree-one and the degree-two building blocks can be generated with less complexities. To generate the PU matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{F}[\mathbf{z}])$, where $\mathbf{z} = (z_1, \ldots, z_r)$,

we employ $N \in \mathbb{N}$ univariate building blocks in each variable. The value of N (typically 2) is fixed and independent of n. Let $\mathbf{C}_{(i-1)N+1}(z_i), \ldots, \mathbf{C}_{iN}(z_i)$ be the PU building blocks in the variable z_i for all $i \in [\, r \,]$. The PU matrix $\mathbf{P}$ is obtained as follows

$$\mathbf{P}(\mathbf{z}) = \prod_{i=1}^{rN} \mathbf{C}_{\sigma(i)}(z_{\lceil \sigma(i)/N \rceil}) \ , \tag{6.39}$$

where $\sigma \in \mathfrak{S}_{rN}$ is a public permutation. We note that since these building blocks do not commute, the order of terms in the above multiplication is important. We employ the method explained in Appendix B for the efficient generation of the PU matrix.

For $\boldsymbol{\varphi}$, we use the structure suggested in (6.34) for the irreducible polynomial χ as in (6.36) (in which the value of ω is public) and the vector polynomial $\boldsymbol{\rho}$ as follows.

$$\rho_i(\mathbf{x}) = \alpha_i x_{r-i+1} + \beta_i \prod_{j=r-i+2}^{n} x_j^{a_{ij}} \qquad \forall i \in [\, r \,] \tag{6.40}$$

Here, $a_{ij} \in \mathbb{N}$ are public exponents and $\alpha_i, \beta_i \in \mathbb{F}^{\times}$, for all $i \in [\, r \,]$, are secret coefficients whose values are obtained from the set $\mathcal{K}$ in Algorithm 6.3. The exponents a_{ij} directly influence the degree of the final public polynomials. As we will explain later, to make sure that some attacks are not applicable on the system, we choose these exponents proportional to the block length n, i.e.,

$$a_{ij} = O(n) \qquad \forall i, j \ . \tag{6.41}$$

As the result, the total degree of the public polynomials becomes proportional to n. Notice that since all the computations are performed in $\mathsf{GF}(2^m)$, all exponents are modulo $2^m - 1$. Hence, if $2^m \leq n$, (6.41) will not have the desired effect. The following fact gives the complexity of constructing $\boldsymbol{\varphi}$.

FACT 6.5.1. *The complexity of constructing $\boldsymbol{\varphi}$ as in (6.34) is $O(r) = O(1)$ since r is constant.*

The next step is composing $\mathbf{P}(\mathbf{z})$ and $\boldsymbol{\varphi}(\mathbf{x})$ to get the matrix polynomial $(\mathbf{P} \circ \boldsymbol{\varphi})(\mathbf{x})$. Let $\mathbf{P}(\mathbf{z}) = [\, p_{ij}(\mathbf{z}) \,]$, where $p_{ij}(\mathbf{z}) = \sum_{\boldsymbol{\alpha} \in \mathcal{C}} p_{ij\boldsymbol{\alpha}} \mathbf{z}^{\boldsymbol{\alpha}} \in \mathbb{F}[\mathbf{z}]$ and $\mathcal{C} \subset \mathbb{Z}^r_{\geq 0}$ is a finite set such that $|\mathcal{C}| = O(1)$ by Fact B.1. To construct $\mathbf{P}(\boldsymbol{\varphi}(\mathbf{x})) = [\sum_{\boldsymbol{\alpha} \in \mathcal{C}} p_{ij\boldsymbol{\alpha}} \boldsymbol{\varphi}^{\boldsymbol{\alpha}}(\mathbf{x})]$, we must compute

$\boldsymbol{\varphi}^{\boldsymbol{\alpha}}(\mathbf{x})$ for all $\boldsymbol{\alpha} \in \mathcal{C}$. Since the exponents $\boldsymbol{\alpha} \in \mathcal{C}$ are independent of n, the complexity of computing $(p_{ij} \circ \boldsymbol{\varphi})(\mathbf{x})$ is $O(|\mathcal{C}|)$. Hence, the total complexity of constructing $\mathbf{P} \circ \boldsymbol{\varphi}$ is $O(|\mathcal{C}|n^2) = O(n^2)$.

Using Fact B.1, the following fact can be stated.

FACT 6.5.2. *Entries of the matrix $\mathbf{P} \circ \boldsymbol{\varphi}$ are multivariate polynomials whose monomials are subsets of a maximal set of monomials. The cardinality of the maximal set is independent of n.*

Having the matrix polynomial $\mathbf{P} \circ \boldsymbol{\varphi}$, an automorphism $\mathbf{t}$ is required to obtain the public vector-polynomial $\boldsymbol{\Psi}$. We suggest the composite automorphism

$$\mathbf{t} = \mathbf{t}_2 \circ \mathbf{t}_1 \ , \tag{6.42}$$

where $\mathbf{t}_1$ and $\mathbf{t}_2$ are tame automorphisms over $\mathbb{F}^n$. If $\mathbf{t}_1 = [\, t_{11}, \ldots, t_{1n} \,]^\dagger$, then

$$t_{1i}(\mathbf{x}) = x_i + \eta_i \prod_{j=1}^{i-1} x_j^{b_{ij}} + \xi_i \qquad \forall i \in [\, n \,] \ , \tag{6.43}$$

where $\eta_2, \ldots, \eta_n \in \mathbb{F}^\times$, $\xi_1, \ldots, \xi_n \in \mathbb{F}$, and $b_{ij} \in \mathbb{N}$ for all $i \in \{\, 2, \ldots, n \,\}$ and $j \in [\, i-1 \,]$. Similarly, if $\mathbf{t}_2 = [\, t_{21}, \ldots, t_{2n} \,]^\dagger$, then

$$t_{2i}(\mathbf{x}) = x_{n-i+1} + \mu_i \prod_{j=2}^{\min(i,K)} x_{n-i+j}^{c_{ij}} \qquad \forall i \in [\, n \,] \ , \tag{6.44}$$

where $\mu_2, \ldots, \mu_n \in \mathbb{F}^\times$, $c_{ij} \in \mathbb{N}$ for all $i \in \{\, 2, \ldots, n \,\}$ and $j \in \{\, 2, \ldots, \min(i,k) \,\}$, and K is a constant such that $K < n$ (a typical value is $K = 5$). The coefficients η_i and ξ_i in (6.43) and μ_i in (6.44) are kept secret and their values are obtained from the set $\mathcal{K}$ in Algorithm 6.3. The exponents b_{ij} and c_{ij} are public. To keep the complexity of the encryption low, we impose the restriction $b_{ij}, c_{ij} \leq B$ for all i and j, where B is a fixed integer independent of the block length n. We note the following important fact.

FACT 6.5.3. *Each entry of $\mathbf{t}$ is a multivariate polynomial that has a constant number of monomials independent of n.*

The complexities of evaluating $\mathbf{t}$ and $\mathbf{t}^{-1}$ are given in the following facts.

FACT 6.5.4. *Complexities of evaluating* $\mathbf{t}$ *and* $\mathbf{t}^{-1}$ *are both* $O(n^2)$.

The next step in generating the OWF $\boldsymbol{\Psi}$ is multiplying $(\mathbf{P} \circ \boldsymbol{\varphi})(\mathbf{x})$ and $\mathbf{t}(\mathbf{x})$ to get the vector polynomial $\boldsymbol{\Psi}_1(\mathbf{x})$ as in (6.25). By Fact 6.5.2 and Fact 6.5.3, the complexity of carrying out this multiplication is $O(n^2)$. The vector polynomial $\boldsymbol{\Psi}_1$ consists of n multivariate polynomials whose number of monomials, given by the following fact, influences the complexity of the encryption.

FACT 6.5.5. *Entries of* $\boldsymbol{\Psi}_1$ *are polynomials whose monomials are subsets of a maximal set of monomials. The cardinality of the maximal set is* $O(n)$.

The final step is generating a unitary matrix $\mathbf{A}$ and multiplying it by the vector polynomial $\boldsymbol{\Psi}_2$ defined in (6.26). As explained in Section 2.2 in Part I, all unitary matrices are generated by multiplying copies of the building block $\mathbf{U}_{\zeta,\mathbf{v}}$ defined in (2.7). To reduce the complexity, we use only one building block for $\mathbf{A}$ with $\zeta = 1$ and $\mathbf{v}$ taken from $\mathcal{K}$. The algorithm in Appendix B can be used to generate $\mathbf{A}$ with complexity $O((n+r)^2) = O(n^2)$. Once we have the unitary matrix $\mathbf{A}$, the final step is performing the multiplication $\mathbf{A}\,\boldsymbol{\Psi}_2$. Entries of the matrix $\mathbf{A}$ are constants, but those of $\boldsymbol{\Psi}_2$ are multivariate polynomials that have $O(n)$ terms by Fact 6.5.5. Hence, the complexity of carrying out the multiplication $\mathbf{A}\,\boldsymbol{\Psi}_2$ is $O(n(n+r)^2) = O(n^3)$. The number of monomials of the entries of $\boldsymbol{\Psi}$ is given in the following fact.

FACT 6.5.6. *Entries of* $\boldsymbol{\Psi}$ *are polynomials whose monomials are subsets of a maximal set of monomials. The cardinality of the maximal set is* $O(n)$.

All the exponents involved in the construction of $\boldsymbol{\Psi}$ are fixed integers except the exponents a_{ij} in (6.40) that are proportional to n. Hence, the following fact can be stated about the total degree of $\boldsymbol{\Psi}$.

FACT 6.5.7. *The total degree of the public polynomials in* $\boldsymbol{\Psi}$ *is proportional to* n.

Complexities of constructing the suggested instance of the PAC are summarized in Table 6.1.

A toy example of the PAC is provided in Appendix C. We note that this is not a practical example of the PAC, and the resulting public-key system is insecure in practice due to small choices for parameters. The purpose of this example is to show how the system is designed and illustrate the structure of public polynomials.

6.5.2 Complexity of the PAC

In this section, we give complexities of the key generation, the encryption, and the decryption in the PAC. Adding up the complexities listed in Table 6.1, we conclude that the total complexity of the public-key generation is $O(n^3)$. The secret key consists of the PU matrix $\mathbf{P}$, the automorphism $\mathbf{t}$, the unitary matrix $\mathbf{A}$, and the constant vector $\mathbf{b}$. By Table 6.1, the total complexity of generating these matrices and vectors is $O(n^2)$.

To compute the complexity of Algorithm 6.1 that is the encryption algorithm, we note that by Fact 6.5.6, the public polynomials $\Psi_1, \ldots, \Psi_{n+r}$ (entries of $\mathbf{\Psi}$) share the same set of monomials. Let $\mathcal{M} \subset \mathbb{Z}_{\geq 0}^n$ be the set of exponents of all monomials. Then

$$\Psi_i(\mathbf{x}) = \sum_{\boldsymbol{\alpha} \in \mathcal{M}} \Psi_{i,\boldsymbol{\alpha}}\, \mathbf{x}^{\boldsymbol{\alpha}} \ , \tag{6.45}$$

where $\Psi_{i,\boldsymbol{\alpha}} \in \mathbb{F}$. Thus, $\mathbf{\Psi}(\mathbf{x})$ has the matrix formulation $\mathbf{\Psi}(\mathbf{x}) = \mathbf{\Phi}\mathbf{X}$, where $\mathbf{\Phi} := [\, \Psi_{i,\boldsymbol{\alpha}} \,]$ is an $(n+r) \times |\mathcal{M}|$ matrix and $\mathbf{X} := [\, \mathbf{x}^{\boldsymbol{\alpha}_1}, \ldots, \mathbf{x}^{\boldsymbol{\alpha}_{|\mathcal{M}|}} \,]^\dagger$ is a vector of length $|\mathcal{M}|$. The complexity of computing $\mathbf{\Phi}\mathbf{X}$ is $|\mathcal{M}|\,(n+r)$. Since $|\mathcal{M}| = O(n)$ by Fact 6.5.6, the total complexity is $O(n^2)$. The complexity of evaluating the vector $\mathbf{X}$ at the plaintext block is $O(n^3)$ by Fact 6.5.7. Hence, the total complexity of the encryption is $O(n^3)$.

For the decryption, Algorithm 6.2 is employed. The complexity of computing $\hat{\mathbf{v}}$ in this

Table 6.1: Complexities of constructing the components of PAC.

	Complexity		*Complexity*		*Complexity*		*Complexity*
$\mathbf{P}$	$O(n^2)$	$\mathbf{P} \circ \boldsymbol{\varphi}$	$O(n^2)$	$\mathbf{\Psi}_1$	$O(n^2)$	$\mathbf{A}\,\mathbf{\Psi}_2$	$O(n^3)$
$\boldsymbol{\varphi}$	$O(1)$	$\mathbf{t}$	$O(n^2)$	$\mathbf{A}$	$O(n^2)$		

algorithm is $O((n+r)^2) = O(n^2)$. Since by Fact B.1 every entry of the PU matrix $\mathbf{P}$ has constant number of monomials (when only degree-one and degree-two building blocks are used), the complexity of computing $\mathbf{P}^{\dagger}(z_1^{-1}, \ldots, z_r^{-1})$ in Algorithm 6.2 is $O(n^2)$. Using Fact 6.5.4, the complexity of computing the plaintext vector $\mathbf{x}$ in this algorithm is $O(n^2)$. Hence, the total complexity of the decryption is $O(n^2)$.

We summarize the complexity of PAC in Table 6.2. The complexities of HFE and RSA public-key schemes are also provided for comparison[5]. The table shows that the computational complexity of the public-key generation and the decryption in the proposed PAC is lower than those in the HFE. Moreover, the PAC outperforms the RSA in terms of key generation, encryption, and decryption. In Table 6.2, m is the number of bits per field element.

6.6 Cryptanalysis of the Instance of the PAC

The entries of the vector polynomial $\mathbf{\Psi}$ are the public information in PAC. To attack this scheme, one approach is solving the system of polynomial equations $y_i = \Psi_i(\mathbf{x})$, $i \in [\, n+r \,]$, for $\mathbf{x}$, where $\mathbf{y} = (y_1, \ldots, y_{n+r})$ is the ciphertext. The other approach is finding the secret key from the public polynomials. In this section, we investigate the vulnerability of PAC to algebraic attacks initially developed for the HFE family. Some of these attacks are quite general and applicable on other schemes. We also investigate the vulnerability of

[5]In RSA, mn is the bit length of the modulus.

Table 6.2: Complexity comparison in the number of binary multiplications.

	Public-Key Generation	*Secret-Key Generation*	*Encryption*	*Decryption*
PAC	$O(m^2n^3)$	$O(m^2n^2)$	$O(m^2n^3)$	$O(m^2n^2)$
HFE	$O(m^2n^4)$	$O(m^2n^2)$	$O(m^2n^3)$	$O(m^2n^2(m+\log n))$
RSA	$O(m^3n^3)$	$O(m^3n^3)$	$O(m^3n^3)$	$O(m^3n^3)$

the PAC for some bad choices of parameters. These attacks follow one of the approaches mentioned above. Our results show that the practical instance of the PAC, introduced in Section 6.5, is resistant to all these attacks. Note that key exhaustive-search has the complexity $|\mathbb{F}|^n \geq 2^{128}$ that is infeasible. Attacks studied in this section are Gröbner basis, univariate polynomial representation, XL, and FXL algorithms, and an attack for small r.

Since the public polynomials of HFE are homogenous, all attacks developed for HFE are specialized for homogenous polynomials. The public polynomials in PAC are not homogenous. However, they can be converted into the homogenous form using a technique employed in algebraic geometry for going from the affine space to the projective space [174]. Let θ_i be the total degree of the public polynomial Ψ_i in the PAC for all $i \in [\, n+r \,]$. Suppose $\theta = \max\{\theta_1, \ldots, \theta_{n+r}\}$. To convert the system of public polynomials into a system of homogenous polynomial equations, replace x_i by X_i/X_0 for all $i \in [\, n \,]$ and multiply through each equation by X_0^θ. The result is the following system of homogenous equations of degree θ that consists of $n+r$ equations in $n+1$ variables $X_0, \ldots, X_n$

$$X_0^\theta y_i = X_0^\theta \, \Psi_i(\tfrac{X_1}{X_0}, \ldots, \tfrac{X_n}{X_0}) \qquad \forall i \in [\, n+r \,] \ . \tag{6.46}$$

By Fact 6.5.7, we note that the total degree of the homogenous polynomials in this system is proportional to n, i.e., $\theta = O(n)$.

6.6.1 Gröbner Basis

As explained in Section A, this is a classical method for solving systems of polynomial equations. Although it can theoretically solve every system of polynomial equations, its complexity is exponential in the number of variables for randomly chosen polynomials. Because of the special form of the public polynomials, computing the Gröbner basis of the HFE-based schemes is feasible using the algorithm F_5 introduced in [79]. As expressed in [80], "A crucial point in the cryptanalysis of HFE is the ability to distinguish a random algebraic system from an algebraic system coming from HFE." As justified in Section 6.4, the public polynomials of the PAC are indistinguishable from a system of randomly chosen

polynomials if Conjecture 6.4.1 holds. Hence, we expect the complexity of computing the Gröbner basis of the public polynomials of the PAC to be exponential in the number of variables. This computation becomes infeasible in the practical instance provided in Section 6.5 in which the number of variables is 16. Moreover, when the total degree of the polynomials is proportional to the number of variables, there does not seem to exist a polynomial time algorithm to compute the Gröbner basis [80]. We note that this is the case in the homogenous form of the public polynomials in the PAC by (6.46).

6.6.2 Univariate-Polynomial Representation of the Public Polynomials

This attack is based on the observation that any system of n multivariate polynomials in n variables over a field $\mathbb{F}$ can be represented as a single sparse univariate polynomial of a special form over an extension field $\mathbb{K}$ of degree n over $\mathbb{F}$. This is summarized in the following lemma that is proved in [116].

LEMMA 6.6.1. *Let $f_i(x_1, \ldots, x_n)$, $i \in [\, n \,]$, be any system of n multivariate polynomials in n variables over $\mathbb{F}$ with the cardinality q. Then, there are coefficients $a_0, \ldots, a_{q^n-1} \in \mathbb{K}$ such that the system of polynomials is equivalent to the univariate polynomial $F(x) = \sum_{i=0}^{q^n-1} a_i x^i$.*

The drawback of this approach is that the number of terms of the equivalent univariate representation $F \in \mathbb{K}[x]$ is exponentially related to the number of variables. However, when the polynomials f_i are homogenous, which is the case in HFE, the polynomial F is sparse. This fact, stated in the following lemma, significantly enhances the attack on the HFE using univariate polynomial representation.

LEMMA 6.6.2. *Let $\mathcal{C}$ be any collection of n homogenous multivariate polynomials of degree θ in n variables over $\mathbb{F}$. Then, the only powers of x that appear in the univariate polynomial representation F over $\mathbb{K}$ are sums of exactly θ (not necessarily distinct) powers of q, i.e., $q^{i_1} + \cdots + q^{i_\theta}$. Hence, the number of nonzero terms and the degree of F are both $O(n^\theta)$.*

To apply the above technique to solve the homogenous form of the public polynomials in the PAC in (6.46), we recall that the degree of the homogenous polynomials θ is proportional

to n. Hence, the degree and the number of nonzero terms of the univariate polynomial representation F are both $O(n^n)$. The complexity of root finding algorithms, e.g., Berlekamp algorithm, is polynomial in the degree of the polynomial [136]. This results in an exponential time algorithm to find the roots of F. Therefore, this approach is less efficient than the exhaustive search.

6.6.3 XL and FXL Algorithms

As explained in Section 5.3.2, these are techniques for solving over-defined systems of polynomial equations. To mount an attack on the HFE scheme using these methods, the equivalent univariate polynomial representation of the public polynomials are obtained using Lemma 6.6.1. By Lemma 6.6.2, it has the form $G(x) = \underline{\mathbf{x}}\,\mathbf{G}\,\underline{\mathbf{x}}^{\dagger}$, where $\mathbf{G} := [\, g_{ij} \,]$ and

$$\underline{\mathbf{x}} := \left[\, x^{q^0}, \ldots, x^{q^{n-1}} \,\right] \ .$$

It is shown in [115] that the cryptanalyst can use this matrix representation to obtain a system of $O(n^2)$ polynomial equations in $O(n)$ variables. One of the algorithms in the XL family can be used to solve this system. Since the homogenous form of the public polynomials of the PAC in (6.46) are not quadratic, their univariate polynomial representation is not quadratic. Hence, it does not have a matrix representation as $G(x)$. Therefore, the attack developed in [115] is not applicable on the PAC.

The adversary may directly apply one of the XL or FXL algorithms to the system of homogenous polynomials (6.46). By (5.20), the minimum number of equations needed is $O(n^\theta)$, where n is the number of variables and θ is the degree of the homogenous equations. In the homogenous form of the public polynomials in PAC, (6.46), we have $\theta = O(n)$. Hence, the number of equations needed is $O(n^{\epsilon n})$ for some $\epsilon > 0$. However, in PAC, the number of equations $n + r \leq 2n$. Therefore, the XL algorithm is unsuccessful in solving the system of public polynomials of the PAC.

To apply the FXL algorithm, the number of recommended guesses is $O(\sqrt{n})$, as explained in Section C, in which case the complexity of the FXL algorithm approximately becomes

$2^{r\sqrt{n}(\log_2 n+8)}$ over $\mathsf{GF}(256)$. For the practical instance of the PAC in which $n = 16$ and $r = 10$, the complexity of the FXL algorithm is approximately 2^{480} that is infeasible.

6.6.4 An Attack for Small r

This attack is applicable on the PAC when r is small, specially $r = 1$, and also when $\mathbf{t} = [\, t_1, \ldots, t_n \,]^\dagger$ in (6.27) is a tame automorphism of the form

$$t_i(\mathbf{x}) = x_i + g_i(x_1, \ldots, x_{i-1}) \qquad \forall i \in [\, n \,] \ , \tag{6.47}$$

where $g_i \in \mathbb{F}[x_1, \ldots, x_{n-1}]$. We briefly describe the attack for $r = 1$. In this case, $\boldsymbol{\varphi}$ is a multivariate polynomial in $\mathbf{x}$, denoted by $\varphi(\mathbf{x})$, i.e.,

$$\varphi(\mathbf{x}) = \chi(\alpha x_1 + \beta \prod_{i=2}^{n} x_i^{a_i}) \ . \tag{6.48}$$

The adversary fixes $x_1, \ldots, x_{n-1}$ and computes the value of $\boldsymbol{\Psi}$ for all $x_n \in \mathbb{F}$. There exists a subset $\mathcal{D} \subset \mathbb{F}$ and a constant $\varphi_0 \in \mathbb{F}$ such that for all $x_n \in \mathcal{D}$, $\varphi(\mathbf{x}) = \varphi_0$. The PU matrix becomes the constant matrix $\mathbf{P}(\varphi_0)$ over $\mathcal{D}$. Because of the special structure of the automorphism $\mathbf{t}$ in (6.47), the values of the polynomials $t_1, \ldots, t_{n-1}$ do not change over $\mathcal{D}$ since they depend only on $x_1, \ldots, x_{n-1}$. The only polynomial that varies over $\mathcal{D}$ is t_n. This implies that $\mathcal{E} := \{\, \boldsymbol{\Psi}(\mathbf{x}) : \, x_n \in \mathcal{D} \,\}$ is a one-dimensional subspace of $\mathbb{F}^{n+1}$. Examination of $\mathcal{E}$ gives the value of the last column of $\mathbf{P}$ up to scaling.

In the next step, the adversary fixes $x_1, \ldots, x_{n-2}$ and computes the value of $\boldsymbol{\Psi}$ for all $(x_{n-1}, x_n) \in \mathbb{F}^2$. Using a similar approach, the adversary can obtain some information about the next-to-the-last column of the PU matrix $\mathbf{P}$. Repeating this process, the adversary is able to obtain useful information about the PU matrix. This attack works for two reasons:

1. The variable x_n appears only in the last entry of the automorphism $\mathbf{t}$. Hence, by fixing $x_1, \ldots, x_{n-1}$, the polynomials $t_1, \ldots, t_{n-1}$ become constant. The practical instance of the PAC, introduced in Section 6.5, does not have this problem. The automorphism $\mathbf{t}$ employed in the practical instance is the composition of two tame automorphisms $\mathbf{t}_1$ and $\mathbf{t}_2$ given in (6.43) and (6.44). By the special structure of these automorphisms, every variable appears in at least K entries of $\mathbf{t}$.

2. In the example given here, $\mathbb{F}[\varphi(\mathbf{x})]$ has the lowest transcendental degree. To avoid such attacks, the value of r should not be small. In general, in order to find $\mathcal{D}$, the adversary must examine the set $\mathbb{F}^r$ that has cardinality $|\mathbb{F}|^r = 2^{8r}$. For the typical choice $r = 10$, the size of this space is 2^{80}. Thus, finding $\mathcal{D}$ becomes infeasible for the adversary.

6.7 Paraunitary Digital Signature Scheme (PDSS)

The general framework explained in Section 6.2 can be used toward the design of signature schemes. Before providing more detail, we define digital signatures. A signature scheme within the public-key framework is defined as a triple of algorithms $(\mathsf{KeyGen}, \mathsf{Sig}, \mathsf{Ver})$ that are explained below [95].

♢ The key generation algorithm KeyGen is a probabilistic, polynomial-time algorithm that on an input $\mathbf{k} \in \mathfrak{K}$ as a security parameter produces the public key $\mathcal{P}$ and the secret key $\mathcal{S}$. Here, $\mathfrak{K}$ is the key space of finite size that has to be large enough to make exhaustive search infeasible.

♢ The signature algorithm Sig is a probabilistic polynomial-time algorithm that is provided with the security parameter $\mathbf{k}$, the secret key $\mathcal{S}$, and a message $\mathbf{y}$. Based on these inputs, the signature algorithm produces as output the signature $\mathbf{z} = \mathsf{Sig}(\mathbf{k}, \mathcal{S}, \mathbf{y})$.

♢ The verification algorithm Ver is a probabilistic polynomial-time algorithm that is provided with the public key $\mathcal{P}$, a signature $\mathbf{z}$, and a message $\mathbf{y}$. This algorithm returns either accept or reject to indicate whether the signature is valid or not.

We say that a digital signature is secure if an adversary who can use the real signer as *an oracle* can not in time polynomial in the size of the public key forge a signature for any message which its signature was not obtained from the real signer.

Based on the mathematical conjecture discussed in Section 6.4, this approach provides a simple relationship between any vector polynomial and those acting as automorphisms over the underlying finite field. This relationship involves multivariate PU matrices. The

advantage of this technique is that if the conjecture is proved to hold, the set of public polynomials in the resulting cryptosystem are indistinguishable from a set of randomly selected multivariate polynomials[6]. Therefore, the security of the designed cryptosystem would be *proved* to be based on the difficulty of solving Problem 4.4.1. In this section, we design such signature scheme. Throughout this section, as in (6.23), we assume $\mathbb{L}_n$ and $\mathbb{P}_n$ are the rings of Laurent and ordinary polynomials, respectively, in the variable vector $\mathbf{x} = (x_1, \ldots, x_n)$ for some arbitrary $n \in \mathbb{N}$.

By (6.24) and assuming Conjecture 6.4.2 holds, for any vector polynomial $\mathbf{f} \in \mathbb{L}_n^n$ and any tame automorphism $\mathbf{t} \in \mathbb{L}_n^n$, there exists a matrix polynomial $\mathbf{P} \in \mathsf{PU}_n(\mathbb{L}_n)$ such that $\mathbf{f}(\mathbf{x}) = \mathbf{P}(\mathbf{x})\,\mathbf{t}(\mathbf{x})$. Assuming $\mathbf{y}$ is the message to be signed and $\mathbf{x}$ is the signature, the objective, in a simple language, is solving $\mathbf{y} = \mathbf{P}(\mathbf{x})\,\mathbf{t}(\mathbf{x})$ for $\mathbf{x}$ by taking advantage of the property of PU matrices and the fact that $\mathbf{t}$ is an automorphism. In other words, we desire to provide an efficiently-computable solution as $\mathbf{x} = \mathbf{t}^{-1}\left(\mathbf{P}^{\dagger}(\mathbf{x})\,\mathbf{y}\right)$. To make the right hand side of this equation independent of $\mathbf{x}$, we take an approach similar to the one used in designing the PAC, i.e., we approximate the PU matrix by composing it with a vector polynomial. To this end, we propose the mapping

$$\begin{array}{rccl} \boldsymbol{\Psi} & : & \mathbb{F}^{n+r} & \longrightarrow & \mathbb{F}^n \\ & & (\mathbf{x}, \mathbf{x}') & \longmapsto & \mathbf{y} = (\mathbf{P} \circ \boldsymbol{\varphi})\,(\mathbf{x}, \mathbf{x}')\,\mathbf{t}(\mathbf{x}) \ , \end{array} \tag{6.49}$$

where $\mathbf{P} \in \mathsf{PU}_n(\mathbb{L}_r)$ for a fixed $r \in \mathbb{N}_{\leq n}$, $\mathbf{x}' = (x'_1, \ldots, x'_r)$, $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{x}') \in \mathbb{L}_{n+r}^r$, and $\mathbf{t} \in \mathbb{L}_n^n$ is a bijection over $\mathbb{F}^n$. Let $\boldsymbol{\varphi}_{\mathbf{x}'}(\mathbf{x})$ and $\boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}')$ denote the polynomial vector $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{x}')$ when $\mathbf{x}'$ and $\mathbf{x}$ are fixed, respectively. The only restriction that we impose on $\boldsymbol{\varphi}$ is that for any $\mathbf{x} \in \mathbb{F}^n$, the polynomial mapping $\boldsymbol{\varphi}_{\mathbf{x}} : \mathbb{F}^r \to \mathbb{F}^r$ must be bijective. The following lemma shows that the proposed mapping satisfies all the properties required for a signature scheme.

LEMMA 6.7.1. *Consider the mapping $\boldsymbol{\Psi}$ in (6.49) with the following specifications:*

(i) $\mathbf{P} \in \mathsf{PU}_r(\mathbb{L}_r)$,

(ii) $\mathbf{t} \in \mathbb{L}_n^n$ *is a bijection over* $\mathbb{F}^n$ *that can be efficiently inverted, and*

[6]We emphasize that similar to Section 6.4, our results on the security of the signature scheme proposed in this section only concern the one-way property of the underlying OWF.

(iii) $\boldsymbol{\varphi} \in \mathbb{L}_{n+r}^{r}$ *such that* $\boldsymbol{\varphi}_{\mathbf{x}} : \mathbb{F}^r \to \mathbb{F}^r$ *is a bijection that can be efficiently inverted.*

Then, the mapping $\boldsymbol{\Psi}$ *is many-to-one with* $\mathcal{R}_{\mathbf{f}} = \mathbb{F}^n$ *that can be efficiently inverted.*

Proof. Let $\mathbf{y} \in \mathbb{F}^n$ be an arbitrary vector. Randomly choose a vector $\mathbf{z} = (z_1, \ldots, z_r) \in (\mathbb{F}^{\times})^r$, and set $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{x}') = \mathbf{z}$. Since $\mathbf{t}$ is an efficiently invertible bijection, the value of $\mathbf{x}$ is uniquely obtained as follows using the inverse of the PU matrix $\mathbf{P}$.

$$\mathbf{x} = \mathbf{t}^{-1}\left(\mathbf{P}^{\dagger}(\mathbf{z})\,\mathbf{y}\right) \tag{6.50}$$

In addition, uniquely obtain the value of $\mathbf{x}'$ from the equation $\boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}') = \mathbf{z}$. Since this procedure is valid for all $\mathbf{y} \in \mathbb{F}^n$, the mapping $\boldsymbol{\Psi}$ is surjective. Moreover, by the presented procedure, the value of $\mathbf{x}$ depends on the random choice for $\mathbf{z}$. Hence, $\boldsymbol{\Psi}$ is a many-to-one function that can be efficiently inverted. ∎

In a practical signature scheme based on this approach, to avoid the division by zero that may happen when employing Laurent polynomials, we restrict the design to the ring of polynomials $\mathbb{P}_n$. In other words, we design a PU matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{P}_r)$, compose it with a polynomial vector $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{x}') \in \mathbb{P}_{n+r}^{r}$, and multiply the result by a bijection $\mathbf{t} \in \mathbb{P}_n^n$. By Lemma 6.7.1, the result is a signature scheme that we call paraunitary digital-signature scheme (PDSS) [62]. The signature generation and verification algorithms of the PDSS are presented in Algorithm 6.4 and Algorithm 6.5, respectively. The PDSS should not be used over the binary field for the same reasons as those explained in Section 6.2 for the PAC.

Since the signature generation depends on the random choice for $\mathbf{z}$, the PDSS is a non-deterministic scheme similar to the ElGamal signature scheme [74]. In other words, there is

---

Algorithm 6.4: PDSS sign.

Input: Message $\mathbf{y} \in \mathbb{F}^n$

Output: Signature $(\mathbf{x}, \mathbf{x}') \in \mathbb{F}^{n+r}$

1. Randomly choose $\mathbf{z} = (z_1, \ldots, z_r) \in (\mathbb{F}^{\times})^r$.
2. $\mathbf{x} \leftarrow \mathbf{t}^{-1}\left(\mathbf{P}^{\dagger}(\mathbf{z})\,\mathbf{y}\right)$, $\mathbf{x}' \leftarrow \boldsymbol{\varphi}_{\mathbf{x}}^{-1}(\mathbf{z})$
3. **return** $(\mathbf{x}, \mathbf{x}')$

Algorithm 6.5: PDSS verify.

Input: Message $\mathbf{y} \in \mathbb{F}^n$ and the signature $(\mathbf{x}, \mathbf{x}') \in \mathbb{F}^{n+r}$

Output: Accept or Reject

1. **if** $\mathbf{y} = \boldsymbol{\Psi}(\mathbf{x}, \mathbf{x}')$ **then return** *Accept*
2. **else return** *Reject*

not a unique signature associated with a message. This is an interesting feature that was not possible in the C^* scheme and its variants. The verification algorithm of the PDSS consists only of evaluating the public polynomial-vector $\boldsymbol{\Psi}$ at the signature. Since the polynomial evaluation can be performed very fast and efficient, the signature verification in the PDSS has the same properties. This feature makes the PDSS very attractive for many applications in which a message is signed only once, but verified many times.

6.7.1 Polynomial Vector φ

Since the PDSS and the PAC share many components, we only briefly explain the mapping $\boldsymbol{\varphi}$ which differs in PDSS. As explained in Section 6.2.2, composing the PU matrix $\mathbf{P}$ with the polynomial vector $\boldsymbol{\varphi}$ has the effect of approximating the PU matrix. Similar to (6.33), we have the chain relation

$$\mathbb{F} \subseteq \mathbb{F}[\boldsymbol{\varphi}_{\mathbf{x}'}(\mathbf{x})] \subseteq \mathbb{F}[\mathbf{x}] \ , \tag{6.51}$$

where $\mathbf{x}' \in \mathbb{F}^r$ is a fixed vector. For $\mathbb{F}\,[\,\boldsymbol{\varphi}_{\mathbf{x}'}(\mathbf{x})\,]$ to achieve its maximum possible transcendence degree, which is r, we design $\boldsymbol{\varphi}_{\mathbf{x}'}(\mathbf{x})$ to be semi-invertible, i.e., after evaluating $\boldsymbol{\varphi}_{\mathbf{x}'}(\mathbf{x})$ at $n-r$ coordinates of $\mathbf{x}$, the resulting r-variate polynomial becomes invertible. To achieve this goal, we suggest using a composition of two tame automorphisms as follows

$$\boldsymbol{\varphi} = \mathbf{s} \circ \boldsymbol{\rho}. \tag{6.52}$$

Here, $\mathbf{s} \in (\mathbb{F}\,[\,\mathbf{x}'\,])^r$ is an automorphism (that can be generated by composing tame automorphisms) and $\boldsymbol{\rho} \in (\mathbb{F}\,[\,\mathbf{x}, \mathbf{x}'\,])^r$ is semi-invertible. If $\boldsymbol{\rho} = [\,\rho_1, \ldots, \rho_r\,]^T$, then it can be

easily verified that the following polynomial vector is semi-invertible.

$$\begin{aligned}\rho_i(\mathbf{x},\mathbf{x}') = \alpha_i x'_{\pi(i)} + \beta_i x_{\sigma(r-i+1)} \\ + g_i(x_{\sigma(r-i+2)}, \ldots, x_{\sigma(n)}, x'_{\pi(1)}, \ldots, x'_{\pi(i-1)}) \qquad \forall i \in [\,r\,] \ .\end{aligned} \tag{6.53}$$

Here, $\sigma \in \mathfrak{S}_n$ and $\pi \in \mathfrak{S}_r$ are public permutations, $\alpha_i, \beta_i \in \mathbb{F}^\times$, and $g_i \in \mathbb{F}[x_{\sigma(r+2-i)}, \ldots, x_{\sigma(n)}, x'_{\pi(1)}, \ldots, x'_{\pi(i-1)}]$ for all $i \in [\,r\,]$. After fixing $\mathbf{x}$, we note that $\boldsymbol{\rho}_{\mathbf{x}}$ turns into a tame automorphism in r variables. Now, assume $\mathbf{x}'$ is fixed. By the definition of $\boldsymbol{\rho}$ in (6.53), it can be easily verified that after its evaluation at $n-r$ arbitrary symbols from $\mathbb{F}$ substituted for $x_{\sigma(r+1)}, \ldots, x_{\sigma(n)}$, $\boldsymbol{\rho}$ becomes invertible. Hence, $\boldsymbol{\rho}_{\mathbf{x}'}$ is semi-invertible. Since $\mathbf{s}$ is an automorphism, $\boldsymbol{\varphi}$ in (6.52) has the same property.

6.7.2 Setup Algorithm

The secret key is a vector $\mathbf{k} \in \mathbb{F}^n$ that is randomly selected by the signer. Using Algorithm 5.2, the secret key is expanded to a set of vectors $\mathcal{K}$ of the same size that are used to determine parameters of the PU building block, coefficients of the bijection $\mathbf{t}$, and the polynomial vector $\boldsymbol{\varphi}$. Algorithm 6.6 is used to generate the public and the secret keys in the PDSS.

6.7.3 A Practical Instance of the PDSS

As in Section 6.5, we suggest a practical instance of the PDSS by specifying components of the scheme. In our design, we follow the same guidelines as those mentioned for the PAC. Briefly, the design criteria are: (1) without knowledge of the secret key, exploiting the structure of the public polynomials in an attack must be infeasible, (2) public polynomials should be sparse, and (3) The automorphism $\mathbf{t}$ must be efficiently invertible.

We construct the matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{F}[\mathbf{z}])$ and the automorphism $\mathbf{t} \in \mathsf{Aut}(\mathbb{F}[\mathbf{x}])$ as in (6.39) and (6.42), respectively. For the vector polynomial $\boldsymbol{\varphi}(\mathbf{x},\mathbf{x}')$, we use the structure suggested in 6.52 with $\mathbf{s} = \mathsf{id}$ (the identity automorphism) and $\boldsymbol{\varphi} = [\,\varphi_1, \ldots, \varphi_r\,]^\dagger$ as follows.

$$\varphi_i\,(\mathbf{x},\mathbf{x}') = \alpha_i x'_i + \beta_i x_{r-i+1} + \gamma_i \prod_{j=r-i+2}^{n} x_j^{a_{ij}} \prod_{j=1}^{i-1} (x'_j)^{b_{ij}} \qquad \forall i \in [\,r\,] \tag{6.54}$$

Algorithm 6.6: PDSS key generation.

Input: Master secret-key $\mathbf{k} \in \mathbb{F}^n$

Public Output: Polynomial vector $\mathbf{\Psi} \in (\mathbb{F}[\mathbf{x}, \mathbf{x}'])^n$

Secret Output: $\mathbf{P}$: an r-variate PU matrix in $\mathsf{PU}_n(\mathbb{F}[\mathbf{z}])$, $\boldsymbol{\varphi}$: a vector polynomial in $(\mathbb{F}[\mathbf{x}, \mathbf{x}'])^r$, $\mathbf{t}$: an automorphism in $\mathsf{Aut}(\mathbb{F}[\mathbf{x}])$.

1. Use Algorithm 5.2 to generate the set $\mathcal{K}$.
2. Generate an r-variate PU matrix $\mathbf{P} \in \mathsf{PU}_n(\mathbb{F}[\mathbf{z}])$ by multiplying elementary building blocks whose parameters are taken from the set $\mathcal{K}$.
3. Choose a vector polynomial $\boldsymbol{\varphi} \in (\mathbb{F}[\mathbf{x}, \mathbf{x}'])^r$ with the following properties: (1) invertible when $\mathbf{x} \in \mathbb{F}^n$ is fixed and (2) semi-invertible when $\mathbf{x}' \in \mathbb{F}^r$ is fixed. Use the vectors in $\mathcal{K}$ as the design parameters. (The composition in (6.52) may be used.)
4. Choose an automorphism $\mathbf{t} \in \mathsf{Aut}(\mathbb{F}[\mathbf{x}])$ using vectors in $\mathcal{K}$ as its coefficients.
5. Construct the vector polynomial $\mathbf{\Psi}(\mathbf{x})$ as in (6.49).

Here, $a_{ij}, b_{ij} \in \mathbb{N}$ are public exponents and $\alpha_i, \beta_i, \gamma_i \in \mathbb{F}^*$ are secret coefficients for all $i \in [\,r\,]$. The vectors in the set $\mathcal{K}$ generated by Algorithm 5.2 are used to specify these coefficients. The exponents a_{ij} and b_{ij}, which are made public, directly influence the degree of the final public polynomials. As we will explain later, to make sure that some attacks are not applicable on the scheme, we choose these exponents proportional to the block length n, i.e.,

$$a_{ij} = O(n) \ , \qquad b_{ij} = O(n) \qquad \forall i, j \ . \tag{6.55}$$

As the result, the total degree of the public polynomials is proportional to n. Complexities of constructing the suggested instance of the PDSS are summarized in Table 6.3.

A toy example of the PDSS is provided in Appendix C. We emphasize that this example is insecure and should not be used in practice. The purpose of this example is to show how the scheme is designed and illustrate the structure of public polynomials.

In Table 6.4, we compare the total complexities of the key generation, signature generation, and verification between the suggested instance of the PDSS and the HFE and RSA used as signature schemes. In this table, m is the number of bits per field element, i.e., the

Table 6.3: Complexities of constructing the components of PDSS.

	Complexity		*Complexity*		*Complexity*
$\mathbf{P}$	$O(n^2)$	$\mathbf{P} \circ \boldsymbol{\varphi}$	$O(n^2)$	$\boldsymbol{\Psi}$	$O(n^2)$
$\boldsymbol{\varphi}$	$O(1)$	$\mathbf{t}$	$O(n^2)$		

Table 6.4: Complexity comparison in the number of binary multiplications.

	Public-Key Generation	*Secret-Key Generation*	*Signature Generation*	*Verification*
PDSS	$O(m^2n^2)$	$O(m^2n^2)$	$O(m^2n^3)$	$O(m^2n^3)$
HFE	$O(m^2n^4)$	$O(m^2n^2)$	$O(m^2n^2(m+\log n))$	$O(m^2n^3)$
RSA	$O(m^3n^3)$	$O(m^3n^3)$	$O(m^3n^3)$	$O(m^3n^3)$

underlying field is $\mathsf{GF}(2^m)$. In RSA, mn is the bit length of the modulus. Table 6.4 reveals that the computational complexity of public-key generation for the proposed PDSS is lower than that in the HFE. Moreover, the PDSS is generally less complex than RSA. We note that the assumption 6.55 may not be necessary for the security of the PDSS. By relaxing this condition, the complexities of the PDSS becomes lower than those in the HFE.

6.8 Summary

Multivariate cryptography is the new trend in information security with the goal of designing highly efficient and fast cryptosystems. The security of systems in this family is based on the difficulty of solving systems of multivariate polynomial equations. In contrast to the traditional approach in which the underlying algebraic structure has very large order, the size of the operating field in multivariate cryptosystems is comparatively much smaller. Hence, these systems are much faster and more efficient. These properties make them suitable for implementing in resource limited environments such as smart cards, personal

digital assistants, cell phones, etc.

Using the successful factorization of all PU matrices in terms of fully parameterized elementary building blocks, we proposed new public-key and digital signature schemes in this chapter. We showed that the computational security of the proposed multivariate schemes is based on the difficulty of solving systems of multivariate polynomial equations given a mathematical conjecture holds. In other words, these systems are proved to be computationally secure if the conjecture is true. Such proof does not exist for any of the currently-in-use public-key cryptosystems. By providing practical instances of the proposed schemes, we showed that their efficiency is comparable to other existing multivariate schemes while maintaining security. In addition, we studied the resistance of our designs against some well known algebraic attacks. Our studies did not reveal any vulnerabilities.

Part III

Security of Wireless Sensor Networks

CHAPTER 7

Key Pre-Distribution

7.1 Background Review

In the era of information technology and with the advent of micro-electro-mechanical systems and low power highly integrated electronic devices, wireless sensor networks (WSNs) are expected to play key roles in many applications such as managing energy plants, logistics and inventory, battlefields, and medical monitoring [9]. A typical sensor network may consist of hundreds to several thousands of sensor nodes that are low cost and battery powered, and have limited computation power and memory. Sensor nodes are either randomly or manually scattered in a field. They form an unattended wireless network that collects information about the field such as temperature, illumination, motion, some chemical material, etc. The collected data is partially aggregated and forwarded to a central processing unit, called the sink, that is responsible for interpreting the data and taking appropriate actions (e.g., sending personnel for precise measurements).

Security is a critical issue when sensor networks are deployed in a hostile environment. An adversary can monitor the traffic, capture or impersonate sensor nodes, and provide misleading information. An essential security primitive, which is a building block for many security services, is pairwise key establishment referred to as *key establishment*. Public-key cryptography provides a complete solution in traditional networks. However, key establishment in sensor networks is far more complicated because of the node constraints and networking features.

Any public-key infrastructure requires a trusted third party to distribute public-key certificates. The sink can take the role of a trusted party. However, because of the short com-

munication range and widespread distribution of sensor nodes, the sink cannot be reached by many nodes. To address this problem, there have been proposals for using symmetric cryptography [72] or identity-based signature schemes [207]. In addition, public-key algorithms are complex and require significant computational power and storage. Sensor nodes have limited memory and computational capabilities. Hence, storing public keys, their corresponding codes, and excusing it in the traditional fashion are not desirable [159]. To solve these problems, some new research have been focused on improving the efficiency of public-key algorithms [92, 91, 194, 18].

We believe that the usage of public key in sensor networks depends on the application. If a sensor network is supposed to simultaneously carry out many tasks and live for a long period of time, then energy preservation is very critical. In such cases, public-key cryptography should be avoided. Another motivation for considering alternatives to public-key cryptography is that key establishment can be used for many security services such as end-to-end confidentiality, data integrity, and entity authenticity. If these were the only security services that are required, then symmetric key cryptography would be very efficient and effective.

An alternative is pre-distributing the keys among the sensor nodes in the network; hence, so called key pre-distribution schemes (KPSs). Although this solution seems feasible at the first glance, its realization has more subtleties compared to the previous schemes due to the massive number of nodes in the network and their resource limitations. A naive approach is using a single key to secure the communication traffic in the entire network. Nevertheless, this approach must be avoided since capturing only one node compromises the entire network to the adversary. Another approach, in a network of size n, is storing $n-1$ pairwise keys in every node, each for communicating with one other node in the network. However, in a typical sensor network with thousands of sensor nodes, the storage requirement for this approach is beyond the memory limitations of sensors. Moreover, not every two sensors are in the communication range of each other. If the neighbor sensors of every sensor in the field are known prior to the deployment, then it suffices to store only

pairwise keys for the communication with neighbor sensors. However, such information are unavailable in typical sensor networks.

7.2 Related Work

The first practical KPS was proposed by Eschenauer and Gligor in [77]. In the Eschenauer-Gligor scheme (EG), a large pool of keys and key IDs are randomly generated off-line by the sink. For every sensor node, a small fraction of keys, called the *key ring*, is randomly drawn from the key pool and loaded into the memory of the sensor along with the corresponding key IDs prior to the network deployment. The size of the key ring is independent of the size of the network and is dictated by the memory limitation of the sensors. After the network deployment, each sensor broadcasts the IDs of the keys in its key ring. Upon receiving such broadcast messages, every node determines the neighbors with which it shares at least a key. Every two nodes at the communication range of each other establish a secure link only if they share at least a key in their key rings. The probability of link establishment is determined by the sizes of the key pool and the key ring. The neighbor nodes that are unable to establish a secure link use other neighbors to find a secure path connecting them through which they exchange a secret key.

A WSN can be considered as a random graph in which vertices are the nodes and the edges are links that the pairs of nodes have established using a link key [109]. In this model, the network is connected if and only if there exists a link or a path connecting every two arbitrary nodes. Using the theory of random graphs, we can study the connectivity properties of the network. There are different models for random graphs among which $\mathcal{G}(n, P_{lk})$ suits studying sensor networks with unlimited communication range for every sensor node. In this model, every two nodes, in a network of size n, establish a link key with probability P_{lk}. Using this model, Erdős and Rényi showed in [76] that the asymptotic

probability of the network connectivity is

$$\begin{aligned} P_c &= \underset{n\to\infty}{\mathsf{Prob}}\Big[\mathcal{G}(n, P_{lk}) \text{ is connected}\Big] \\ &= e^{-e^{-c}} \ , \end{aligned} \tag{7.1}$$

where $c \geq 0$ is a real constant satisfying the equation $P_{lk} = (\ln n + c)/n$. From here, the probability of the link establishment for a desired probability of network connectivity is

$$P_{lk} = \frac{\ln\left(-n/\ln P_c\right)}{n} \ . \tag{7.2}$$

Therefore, the expected degree of a node is $d = P_{lk}\,(n-1)$.

In reality, the communication range of every sensor node is limited. Hence, the number of neighbors of every sensor node with which it can establish a link is limited to $n' \ll n$. Given n', the required probability of the link establishment is

$$\begin{aligned} p_{req} &= \frac{d}{n'} \\ &= \frac{(n-1)\ln\left(-n/\ln P_c\right)}{nn'} \gg P_{lk} \ . \end{aligned} \tag{7.3}$$

Considering the fact that any two nodes that share a key in their key rings can establish a link, the actual probability of the link establishment is

$$p_{act} = 1 - \frac{\binom{P}{r}\binom{P-r}{r}}{\binom{P}{r}^2} \tag{7.4a}$$

$$\approx 1 - \frac{\left(1-\frac{r}{P_{lk}}\right)^{2\left(P-r+\frac{1}{2}\right)}}{\left(1-\frac{2r}{P_{lk}}\right)^{\left(P-2r+\frac{1}{2}\right)}} \ , \tag{7.4b}$$

where r and P are the sizes of the key ring and the key pool, respectively. The size of the key pool is obtained from (7.3) and (7.4b) by setting $p_{act} \geq p_{req}$.

Example 7.2.1. *Consider a network of size* $n = 10{,}000$ *with the connectivity probability* $P_c = 0.99999$. *Assume by the transmission range of every sensor node, each one has* $n' = 60$ *neighbors on the average. Let the memory size of every sensor be* 200. *Thus, the size of the key ring must be* $r = 200$. *The average node-degree turns out to be* $d = 21$. *The required probability of link establishment is* $p_{req} = 0.3454$ *by (7.3). This gives a key pool of size* $P = 94{,}616$.

A WSN might be deployed in a hostile environment in which sensor nodes are vulnerable to the physical capture by an adversary. To keep down the cost of the sensors, usually they are not tamper resistant [6, 7]. Therefore, upon the capture of a sensor node, the adversary reads all the information stored in it. However, this information might be used in another part of the network to establish a secure link between two non-captured nodes. In this situation, that link is regarded as compromised. Upon detecting a captured node, the sink broadcasts a key-revocation message to all the nodes to delete the compromised keys. The resilience of the network against the node capture refers to the probability of the link compromise amongst non-captured nodes in the presence an adversary who physically captures sensor nodes. Let x be the number of captured nodes in the network. The probability that none of the keys in the key ring of a sensor node is compromised is $1 - \frac{r}{P}$. Hence, the probability of the link compromise is

$$P_{lkc} = 1 - \left(1 - \frac{r}{P}\right)^x \quad . \tag{7.5}$$

The number of functional and non-malicious sensor nodes in the network decreases in time due to either dying because of consuming all their energy or being captured by an adversary. Hence, in order to keep the network connected and functional, new sensor nodes must be added to the network. The newly added nodes must be able to establish secure links with existing nodes in the network. A KPS that supports the addition of new nodes to the network is called *scalable*. The EG is scalable up to a maximum size of the network.

7.2.1 q-Composite Scheme

In EG, any two neighboring nodes that share a single common key establish a secure link. A modification of this scheme is proposed in [32] in which q common keys ($q > 1$) are required to establish a link. Thus, it is called the q-composite scheme (qComp). Let $k_1, \ldots, k_q$ be all the common keys where $q' \geq q$. The final secret key between the nodes is $k = \chi(k_1, \ldots, k_q)$, where χ is a function symmetric in the order of its input arguments. Increasing the amount of the key overlap required to establish a link improves the resilience of the network against

the node capture. The disadvantage of this technique is that since every two nodes must share at least q keys in their key rings with a certain probability, the adversary compromises more keys by capturing less nodes comparing to EG. Therefore, the probability of the link compromise in the qComp is less than that in the EG for a small number of captured nodes. However, as the number of captured nodes increases, this probability increases in the qComp. A procedure similar to the one employed in the EG is used here to calculate the size of the key pool. The actual probability of the link establishment in the qComp [32]

$$p_{act} = 1 - \sum_{i=0}^{q-1} p(i) \ , \tag{7.6}$$

where

$$p(i) = \frac{\binom{P}{i}\binom{P-i}{2(r-i)}\binom{2(r-i)}{r-i}}{\binom{P}{r}^2} \ , \qquad 0 \leq i \leq r \tag{7.7}$$

is the probability that any two sensor nodes have exactly i shared keys in their key rings. In (7.7), P and r are the sizes of the key pool and the key ring, respectively. The size of the key pool is obtained by setting $p_{act} \geq p_{req}$, where p_{req} is obtained from (7.3).

EXAMPLE 7.2.2. *Consider the network in Example 7.2.1. Using the qComp $q = 2$, the size of the key pool must be $P =$ 47,910. However, it is $P =$ 94,616 in EG as calculated in Example 7.2.1. The reduction in the size of the key pool increases the number of keys shared between any two nodes.*

The probability of the link compromise in the qComp is [32]

$$P_{lc} = \sum_{i=q}^{r} \left(1 - \left(1 - \frac{r}{P}\right)^x\right)^i \frac{p(i)}{p_{act}} \tag{7.8}$$

in which x is the number of captured nodes and $p(i)$ and p_{act} are given in (7.7) and (7.6), respectively.

7.2.2 Blom's Scheme

The resilience of the previous KPSs against the node capture gradually decreases by increasing the fraction of the captured nodes. In some applications, when the fraction of

captured nodes does not increase beyond some threshold, it is desirable to have perfect secrecy before reaching that threshold. A KPS that offers this feature is called a *threshold schemes*. In [20], Blom proposed a threshold scheme for ad hoc networks using maximum distance separable (MDS) codes. The Blom's scheme allows any pair of users in a network to compute a secret pairwise key (with the underlying assumption that the communication range of the sensor nodes is infinite). In addition, prior to capturing at most t nodes, the scheme is perfectly secure. However, once at least $t+1$ nodes are captured, the security completely breaks down. Hence, the scheme is called t-secure. In general, a d-conference t-secure scheme is defined as follows.

DEFINITION 7.2.1 (d-Conference t-Secure Scheme). *Let $\mathcal{U}$ be a set of n users and $d, t \in \mathbb{N}$ such that $d \leq n$. A KPS for $\mathcal{U}$, initialized by a server, is d-conference t-secure if and only if without the involvement of the server:*

1. *every subset $\mathcal{V} \subseteq \mathcal{U}$ of $|\mathcal{V}| = d$ users can compute a group key by cooperating with each other (e.g., exchanging their IDs).*
2. *the coalition of every set $\mathcal{A} \subset \mathcal{U} \setminus \mathcal{V}$ of $|\mathcal{A}| \leq t$ users reveals no information about the group key established by the d users in $\mathcal{V}$.*

Based on this definition, the Blom's scheme is 2-conference t-secure.

To explain the Blom's threshold scheme, consider a network of size n. The sink designs a matrix $\mathbf{G} \in \mathsf{M}_{t+1,n}(\mathbb{F}_q)$, where q is a power of a prime, as the generator matrix of a $(n, t+1)$ linear code over $\mathbb{F}_q$ [197]. The matrix is designed to be full rank that is equivalent to the linear code being MDS [197]. This matrix is made public to everybody including all the users and the adversary. In addition, the sink secretly generates a symmetric matrix $\mathbf{D} \in \mathsf{M}_{t+1}(\mathbb{F}_q)$ and computes the matrix $\mathbf{A} = (\mathbf{DG})^{\dagger} \in \mathsf{M}_{n,t+1}(\mathbb{F}_q)$. Prior to the network deployment, the sink stores the i-th row of the matrix $\mathbf{A}$ and the i-th column of the matrix $\mathbf{G}$ in the memory of the i-th user for all $i \in [n]$. It can be easily verified that the matrix $\mathbf{K} = \mathbf{AG} \in \mathsf{M}_n(\mathbb{F}_q)$ is symmetric since the matrix $\mathbf{D}$ has the same property. After the deployment, the users i and j who want to communicate exchange the columns of the

matrix $\mathbf{G}$ that each one is storing. After the exchange, both users have the i-th and the j-th column of $\mathbf{G}$. Using this information, user i is able to compute k_{ij}, the (i, j) entry of the matrix $\mathbf{K} = [\, k_{ij} \,]$. Similarly, user j is able to compute k_{ji}. Since the matrix $\mathbf{K}$ is symmetric, $k_{ij} = k_{ji}$. Thus, the two users have calculated a common pairwise key. As an example of the generator matrix $\mathbf{G} = [\, g_{ij} \,]$, consider the matrix with entries $g_{ij} = \varepsilon^{(i-1)j}$, for all $i \in [\, t + 1 \,]$ and $j \in [\, n \,]$, in which $\varepsilon \in \mathbb{F}_q$ is a primitive element and $n < q$. This is a Vandermonde matrix, which is full rank. Because of its special structure, an arbitrary user $j \in [\, n \,]$ only stores ε^j instead of the entire j-th column of $\mathbf{G}$.

Since the underlying linear code is MDS, the minimum Hamming distance between any two codewords is $n - t$ [197]. Thus, a codeword is uniquely determined by at least $t + 1$ elements. Since the rows of the matrix $\mathbf{K}$ are, in fact, codewords, the coalition of at least $t+1$ users is required to determine the pairwise key between a pair of other users. Therefore, the Blom's scheme is t-secure.

7.2.3 Du et al.'s Threshold Scheme

Proposed in [71], the Du scheme (Du) is, in fact, a randomized and extended version of the Blom's scheme. In this scheme, a *key space* is a tuple $(\mathbf{D}, \mathbf{G})$, where the matrices $\mathbf{D}$ and $\mathbf{G}$ are as defined in the Blom's scheme. In Du, every node is identified by an ID $i \in [\, n \,]$. The sink generates a Vandermonde matrix $\mathbf{G}$ using a primitive element $\varepsilon \in \mathbb{F}_q$. This matrix is regarded as public information. In addition, the sink secretly generates ω symmetric matrices $\mathbf{D}_i \in \mathsf{M}_{t+1}(\mathbb{F}_q)$ and calculates the corresponding matrices $\mathbf{A}_i = (\mathbf{D}_i \mathbf{G})^T$ for all $i \in [\, \omega \,]$. For every node i, a random index subset $\mathcal{I} \subset [\, \omega \,]$ of cardinality $|\mathcal{I}| = \tau$ is selected, where $\tau < \omega$ is a fixed design parameter of the scheme. For all $j \in \mathcal{I}$, the i-th row of the matrix $\mathbf{A}_j$ is stored in the memory of the node i along with ε^i prior to the network deployment. In other words, the node i *picks* the key spaces $(\mathbf{D}_j, \mathbf{G})$ for all $j \in \mathcal{I}$. By the Blom's scheme, every two nodes that pick the same key space are able to find a common secret key.

The required storage memory in the Du is $M = \tau(t + 1) + 1$. Hence, given M and τ as

the sizes of the sensor memory and key ring, respectively, the security threshold must be $t = \left\lfloor \frac{M-1}{\tau} \right\rfloor - 1$.

After the deployment, every node $i \in [\, n \,]$ broadcasts its ID, the indices of the spaces it has picked, and ε^i. Assume after receiving such broadcast messages, two neighbor nodes have discovered that they have a key space in common. Using Blom's scheme, these nodes can compute a common pairwise key. Similar to previous schemes, two neighbor nodes that do not have any key spaces in common can discover a key path through which they exchange a secret key.

To determine the total number of key spaces ω, a procedure similar to the previous schemes is employed. The required probability of link establishment for a desired probability of network connectivity is obtained from (7.3). The actual probability of the link establishment in Du is [71]

$$p_{act} = 1 - \frac{\binom{\omega}{\tau}\binom{\omega-\tau}{\tau}}{\binom{\omega}{\tau}^2} \quad . \tag{7.9}$$

The value of τ is dictated by the memory constraint of sensor nodes. The total number of key spaces is calculated by setting $p_{act} \geq p_{req}$.

EXAMPLE 7.2.3. *Consider the network in Example 7.2.1 in which the size of the sensor memory is $M = 200$. Assuming $\tau = 4$, the security threshold must be $t = 48$. The number of key spaces must be $\omega = 42$.*

In Du, the probability of link compromise is [71]

$$P_{lc} = \sum_{i=t+1}^{x} \binom{x}{i} \left(\frac{\tau}{\omega}\right)^i \left(1 - \frac{\tau}{\omega}\right)^{x-i} \quad , \tag{7.10}$$

where x is the number of captured nodes.

7.2.4 Liu's Polynomial-Based Schemes

Schemes proposed in [127] are other examples of threshold schemes. In these schemes, symmetric bivariate polynomials are used as a mechanism to calculate pairwise keys. The basic idea of this mechanism was first proposed in [19] and then generalized and further studied in [22].

To explain the polynomial-based KPS proposed in [22], consider a network with the user set $\mathcal{U} = \{ U_0, \ldots, U_{n-1} \}$ in which every user has an ID that is an integer in (n). The server generates a symmetric polynomial $f(x_0, \ldots, x_{d-1})$ in $d \leq n$ variables of degree t in each variable with coefficients from the finite field $\mathbb{F}_p$ for some prime integer $p \geq n$. The polynomial f is symmetric in the following sense

$$f(x_{\sigma(0)}, \ldots, x_{\sigma(d-1)}) = f(x_0, \ldots, x_{d-1}) \qquad \forall \sigma \in \mathfrak{S}_d \ . \tag{7.11}$$

The server assigns the coefficients of the polynomial $f_i(x_1, \ldots, x_{d-1}) \coloneqq f(i, x_1, \ldots, x_{d-1})$ to user U_i for all $i \in (n)$. Since each polynomial f_i has at most $\binom{t+d-1}{d-1}$ monomials, the maximum memory storage requirement for each user is $\binom{t+d-1}{d-1} \log_2 p$ bits. This scheme is optimal in the sense that the amount of information stored in each user is minimal [22].

Any set of at least d users is able to compute a common key using the polynomials in their memories. To see how, consider the set of users $\{ U_i : i \in I \}$, where $I \subseteq (n)$ is an arbitrary subset of size $|I| = d$. The users in this set, first, exchange their IDs. Then, each user U_i evaluates its polynomial f_i at $(x_1, \ldots, x_{d-1}) = I\langle i \rangle$ (see Section subsec:general notation for this notation). In light of (7.11), it is easily verified that the group key is $k_I = f(I)$. Hence, this scheme is d-conference t-secure. The following example shows how this idea works.

A Random Polynomial-Based Scheme

An amalgamation of the EG and the polynomial-based threshold scheme of [22] is proposed in [127], which we refer to as the random polynomial-based scheme (RPB). In this scheme, prior to the network deployment, the sink generates a pool $\{ f_i(x, y) \in \mathbb{F}_p[x, y] : i \in (P) \}$ of P symmetric bivariate polynomials each of degree t in both variables over the finite field $\mathbb{F}_p$ in which p is a prime integer such that $p \geq n$. For every sensor node $i \in (n)$, the sink randomly selects a subset $I \subset (P)$ of size $|I| = \tau$. The sink stores the coefficients of the univariate polynomial $f_j(i, y)$, for all $j \in I$, in the memory of this sensor. The set univariate polynomials stored in every node is called the *polynomial ring*. Using polynomials in their rings, neighbor nodes discover common polynomials and calculate common key using a

method similar to the one in EG. The RPB is a generalized version of the EG in the sense that when the degrees of all polynomials t is zero, the polynomial ring becomes the key ring. Another observation is that the RPB is 2-conference t-secure by Definition 7.2.1.

In the RPB, the required storage memory is $M = \tau(t+1)$. Hence, given M and τ as the sizes of the sensor memory and the polynomial ring, respectively, the security threshold must be $t = \left\lfloor \frac{M}{\tau} \right\rfloor - 1$.

The actual probability of the link establishment in RPB is [127]

$$p_{act} = 1 - \prod_{i=0}^{\tau-1} \frac{P-\tau-i}{P-i} \quad . \tag{7.12}$$

The algorithm for calculating the size of the polynomial pool is similar to the previous schemes.

EXAMPLE 7.2.4. *Consider the network in Example 7.2.1 in which the size of the sensor memory is $M = 200$. Assuming $\tau = 4$, the security threshold must be $t = 49$. The size of the polynomial pool must be $P = 42$.*

To compromise a link in RPB, one of the bivariate polynomials must be recovered using its shares distributed in the network. The probability of recovering a bivariate polynomial is

$$P_{pc} = 1 - \sum_{i=0}^{t} P_{pc}(i) \tag{7.13}$$

where $P_{pc}(i)$ is the probability of compromising i shares of any polynomial, i.e.,

$$P_{pc}(i) = \binom{x}{i} \left(\frac{\tau}{P}\right)^i \left(1 - \frac{\tau}{P}\right)^{x-i} \quad . \tag{7.14}$$

Here, x is the number of captured nodes in the entire network.

B Grid-Based Scheme

The RPB is randomized in the sense that the polynomial ring is randomly selected from the polynomial pool. Consequently, when the transmission range of all sensor nodes is unlimited (a theoretical assumption), it is not guaranteed that every two sensor nodes can establish a key. To remedy this problem, a grid-based scheme (GB) is proposed in [127]

that takes a deterministic approach. In this scheme, for a network of given maximum size n, a two-dimensional grid $(m)^2$ is virtually generated in which $m := \lceil \sqrt{n} \rceil$. This grid is only used as a tool to assign IDs to the nodes and distribute shares of polynomials to them. There is no relationship between the physical locations of the nodes in the field and the points on the grid. The points on the grid are randomly assigned to the sensor nodes as their IDs. Hence, the ID of every node is a tuple $(i, j) \in (m)^2$. In the setup phase, the sink generates a pool of $2m$ symmetric bivariate polynomials $f_i^r(x, y), f_i^c(x, y) \in \mathbb{F}_p[x, y]$ for all $i \in (m)$. As shown in Figure 7.1, each row j in the grid is associated with a polynomial $f_j^r(x, y)$ and each column i is associated with a polynomial $f_i^c(x, y)$. For the node at the coordinate $(i, j) \in (m)^2$, the sink pre-loads its ID and the shares of the two polynomials $f_i^c(x, y)$ and $f_j^r(x, y)$, i.e., the coefficients of the univariate polynomials $f_i^c(j, y)$ and $f_j^r(i, y)$. Therefore, the storage-memory requirement for every sensor node is $M = 2(t + 1)$. Given M as the size of the sensor memory, the security threshold must be $t = \left\lfloor \frac{M}{2} \right\rfloor - 1$.

After the network deployment, nodes broadcast only their IDs since this information is sufficient to determine whether two neighbor nodes share a polynomial or not. Every two nodes that lie on the same horizontal or vertical line on the grid can establish a link key. For example, consider the nodes (i, j) and (i, j') in Figure 7.1. Theses nodes can establish

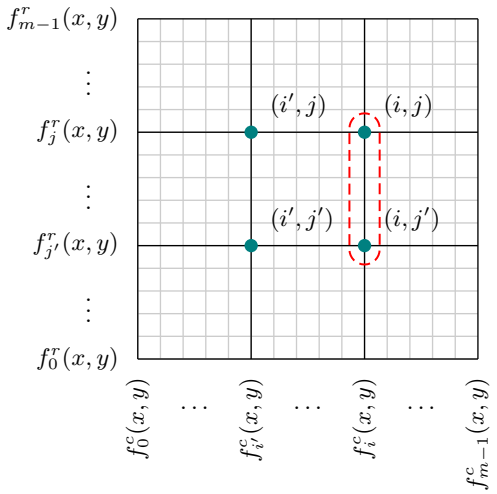


Figure 7.1: Two-dimensional grid.

the link key $f_i^c(j,j') = f_i^c(j',j)$. Similarly, the nodes (i,j) and (i',j) that lie on the same vertical line can establish the direct key $f_j^r(i,i') = f_j^r(i',i)$. Hence, the probability of the link-key establishment in the GB is

$$P_{lk} \lessapprox \frac{2(m-1)}{n-1} \ . \tag{7.15}$$

We note that since GB is not a probabilistic scheme, unlike previous schemes, the total number of bivariate polynomials is independent of the required probability of network connectivity.

The reconstruction of a bivariate polynomial in which both variables have degree t requires at least $t+1$ univariate shares. Therefore, the probability of the link-key compromise (equivalently the probability of compromising a polynomial) is [127]

$$P_{lkc} = 1 - \sum_{i=0}^{t} \binom{m}{i} p_{nc}^i (1-p_{nc})^{m-i} \ , \tag{7.16}$$

where p_{nc} is the faction of captured nodes in the entire network.

C Hypercube-Based Scheme

The hypercube-based scheme (HB) of [127] is a generalization of the GB to higher dimensions. This scheme is designed along the guidelines of GB. The difference is that in the setup phase, the sink generates a d-dimensional hypercube $(\,m\,)^d$, where $m = \lceil \sqrt[d]{n} \rceil$. The ID of every sensor is a d-tuple $I = (i_0, \ldots, i_{d-1}) \in (\,m\,)^d$. The polynomial pool consists of dm^{d-1} symmetric bivariate polynomials $f^j_{I\langle j \rangle}(x,y) \in \mathbb{F}_p[x,y]$, where $j \in (\,d\,)$ and $I \in (\,m\,)^d$. Prior to the network deployment, the sensor node I is loaded with d univariate polynomials $f^0_{I\langle 0 \rangle}(i_0,y), \ldots, f^{d-1}_{I\langle d-1 \rangle}(i_{d-1},y)$. Therefore, the required storage-memory is $M = d(t+1)$. Similar to GB, given M as the size of the sensor memory and d as the dimensionality, the security threshold must be $t = \left\lfloor \frac{M}{d} \right\rfloor - 1$.

In the polynomial-share discovery phase, every two nodes with IDs at the unit Hamming distance of each other can establish a link key. For example, consider the two nodes:

$$I = (i_0, \ldots, i_{j-1}, i_j, i_{j+1}, \ldots, i_{d-1}) \tag{7.17a}$$

$$I' = (i_0, \ldots, i_{j-1}, i'_j, i_{j+1}, \ldots, i_{d-1}) \tag{7.17b}$$

for some $j \in (d)$. These two nodes can establish the link key $f^j_{I\langle j\rangle}(i_j, i'_j) = f^j_{I'\langle j\rangle}(i'_j, i_j)$ since $I\langle j\rangle = I'\langle j\rangle$. Therefore, the probability of the link-key establishment is

$$P_{lkc} \lessapprox \frac{d(m-1)}{n-1} \quad . \tag{7.18}$$

The probability of the link-key compromise in HB is the same as that in the GB given by (7.16).

The main difference between the HB and the GB is that the former provides a better resiliency against the node capture. To see why, we note that to compromise a polynomial, at least $t+1$ shares of that polynomial are required whereas there are only m such shares in the entire network. if $m < t+1$, there are not enough shares for any polynomial to be compromised. Hence, the network becomes perfectly secure against the random attack. Let m_{GB} and m_{HB} be the values of m in the GB and the HB, respectively. We have

$$m_{\mathsf{HB}} = \lceil \sqrt[d]{n} \rceil \leq \lceil \sqrt{n} \rceil = m_{\mathsf{GB}}$$

since $d \geq 2$. Hence, HB provides a better chance for perfect secrecy for a fixed degree t for bivariate polynomials.

7.2.5 Location-Aware KPSs

The first location-aware KPS for wireless sensor networks is proposed in [70]. In this scheme, the sensor nodes are partitioned into groups of almost equal size. The groups of sensors are deployed on the points of a two-dimensional grid on the field. Although the exact location of every sensor is unknown, the expected location of every group is known. The scheme proposed in [70] takes advantage of the expected location information. Unfortunately, it does not provide full connectivity and high resiliency against the node capture. A deterministic location-aware KPS based on bivariate polynomials is proposed in [126]. In this scheme, that we refer to as location-aware grid-based scheme (LA-GB), the deployment field is divided into non-overlapping square cells, and every cell is assigned a unique ID. The sensor nodes are uniformly distributed in the field. The shares of randomly-generated symmetric

bivariate polynomials are distributed to the nodes based on the ID of the cell in which the nodes reside. Using these shares, every two sensors within a cell and in adjacent cells can establish common keys. In this chapter, we will compare this scheme with our proposed location-aware scheme in terms of the resiliency against the node capture.

7.3 Multivariate Key Pre-Distribution Scheme

In this section, we introduce the multivariate key pre-distribution scheme (MKPS) [59, 64, 61, 53]. In the following, This scheme consists of two phases: setup and link-key establishment. The first phase is performed by the sink prior to the network deployment. In this phase, the sink generates a virtual hypercube in the d-dimensional space. This is the smallest hypercube that fits a maximum number of nodes foreseen to be deployed in the network. The points inside the hypercube are uniquely assigned to the sensor nodes. There is no relationship between the hypercube and the actual placement of the nodes in the field after the deployment. The hypercube is only used as a tool to assign IDs to the nodes. The sink generates a symmetric d-variate polynomial for every line parallel to one of the axis lines. Since every point in the d-dimensional space is at the intersection of d lines each parallel to one of the axis lines, d polynomials are associated to every node. Every one of these polynomials is evaluated at $d-1$ variables determined by the node ID. The result is d univariate polynomials that are stored in the node memory.

The link-key establishment phase of the protocol takes place in the field after the nodes are randomly deployed. In this phase, every node broadcasts its own ID. Every two neighbor nodes with IDs at the Hamming distance of one from each other establish exactly $d-1$ common keys by evaluating their univariate polynomials. The final link key is a symmetric combination of the common keys.

We also propose an optimization procedure to choose the optimal dimension d. In contrast to previous schemes that consider only the network security, the optimality criteria in our scheme are the probability of the link-key compromise and the probability of the link-

key establishment. This consideration provides robustness to the designer to adjust the network properties according to the desired application because similar to all other key pre-distribution schemes, increasing one of these metrics decreases the other. We also include a complete evaluation of our proposed scheme. The evaluation metrics are the network connectivity, the resiliency of the network against the node capture, the amount of storage memory required by every sensor node, and the communication overhead during the key establishment. In order to study the connectivity of the MKPS scheme when the dimension-optimization procedure is employed, we employ the random-graph model of the network. This model helps to determine when the network has a giant component that is a connected subgraph containing almost all the nodes except a few isolated ones. In our scheme, the existence of the giant component is considered sufficient for the healthy operation of the network as suggested in [106]. The theoretical analysis of the resiliency of our scheme against the node capture reveals significant improvement over the previous schemes due to the use of multivariate polynomials. In addition, the storage memory required to use the MKPS scheme is at most the same as that in the HB scheme of [127]. We will show that the expected communication-overhead in our scheme is lower than that in previously proposed schemes.

To facilitate future references, frequently used notations are listed below with their meanings.

n	Total number of nodes in the network
d	Dimension of the hypercube
m	$= \lceil \sqrt[d]{n} \rceil$
I	Node ID (a d-tuple)
t	Degree of the multivariate polynomials in every variable
M	Maximum size of the storage memory in every node
$k_{I,I',\ell}$	ℓ-th key common between the nodes I and I'
$k_{I,I'}$	Final link key established between I and I'

P_{lk}	Average probability of link-key establishment
β	Size of the giant component of the network normalized to the network size
λ	Security threshold of the MKPS
P_{pr}	Probability of the polynomial recovery
P_{lkc}	Probability of the link-key compromise
p_τ	Node-capture tolerance
d_{opt}	Optimal dimension
d_{wst}	Worst dimension
γ_1, γ_2	Thresholds used in the dimension-optimization procedure

In the following, we explain the two phases of MKPS in detail.

7.3.1 Setup

In this phase, the sink uniquely assigns IDs to sensor nodes. A node ID is a vector $I = (i_0, \ldots, i_{d-1})$ of length d, where $i_0, \ldots, i_{d-1}$ are nonnegative integers. Let $\mathcal{I}$ be the set of all n IDs. Then, $[m-1]^d \subsetneq \mathcal{I} \subseteq [m]^d$, where $m := \lceil \sqrt[d]{n} \rceil$. The details of the ID-assignment algorithm are provided in Appendix D. Since every node is assigned a unique ID, node-to-node authentication is available in our scheme. This feature enables every node to ascertain the identity of the nodes that it is communicating with [32]. We note that this important feature is missing in the EG and the qComp.

In addition, the sink randomly and independently generates dm symmetric d-variate polynomials $f_i^j(x_0, \ldots, x_{d-1})$ for all $i \in [m]$ and all $j \in [d]$. The coefficients of all these polynomials belong to the finite field $\mathbb{F}$. Moreover, all polynomials have degree t in every variable. For a node with ID $I = (i_0 \ldots, i_{d-1}) \in \mathcal{I}$, the sink constructs the set

$$\mathfrak{F}_I := \left\{ f_{i_j}^j(I\langle j \rangle, x_{d-1}) \in \mathbb{F}[x_{d-1}] \ : \ \forall j \in [d] \right\} \tag{7.19}$$

consisting of d univariate polynomials. Prior to the network deployment, the sink stores the coefficients of all d polynomials in $\mathfrak{F}_I$ in the node I along with its ID. We note that the

virtual hypercube $\mathcal{I}$ has no relationship with the actual placement of the nodes in the field. It is only used as a tool to assign IDs to the nodes and distribute shares of polynomials.

Since $|\mathfrak{F}_I| = d$, the amount of memory required to store all the polynomials in $\mathfrak{F}_I$ is $M = d(t+1)$ in terms of the number of elements in $\mathbb{F}$. The value of M is determined by the memory limitation of the sensor nodes, and d is a design parameter that is either arbitrarily chosen or optimized by the designer. Hence, given M and d, the value of t is

$$t = \lfloor M/d - 1 \rfloor \quad . \tag{7.20}$$

Sets of polynomials assigned to the nodes are deterministically selected in our scheme based on their IDs. Hence, from this view point, our scheme is deterministic comparing to the EG and the qComp in which the key rings are randomly selected from a key pool. However, our scheme appears random in terms of the scattering of the nodes after the network deployment. The following example demonstrates how the polynomial set $\mathfrak{F}_I$ is constructed.

EXAMPLE 7.3.1. *Let $d = 3$. For a node with ID $I = (i_0, i_1, i_2)$, the set $\mathfrak{F}_I$ is equal to $\{ f^0_{i_0}(i_1, i_2, x_2),\ f^1_{i_1}(i_0, i_2, x_2), f^2_{i_2}(i_0, i_1, x_2) \}$.*

7.3.2 Link-Key Establishment

This phase takes place after the deployment of the network in the field. In this phase, every two nodes at the Hamming distance of one from each other establish some common keys. Consider the two nodes I and I' as in (7.17) with Hamming distance one. We note that the IDs of these nodes are different only at the j-th coordinate. By the symmetry property of the polynomials, these nodes can establish the following $d-1$ common keys.

$$k_{I,I',\ell} = f^\ell_{i_\ell}(I\langle j, \ell \rangle, i_j, i'_j) = f^\ell_{i_\ell}(I'\langle j, \ell \rangle, i'_j, i_j) \qquad \forall \ell \in [d] \setminus \{ j \} \tag{7.21}$$

We note that using d-variate polynomials in our scheme has created the situation that every two nodes at Hamming distance of one from each other can establish exactly $d-1$ common keys. Hence, our scheme is in some sense $(d-1)$ composite. As we will show later, this feature greatly lowers the probability of link-key compromise. Fortunately, this

feature is obtained for free without any payoffs. The qComp possesses the same property by requiring that every two nodes share at least q keys in their key rings to establish a direct key. Although this restriction decreases the probability of the link-key compromise for small numbers of captured nodes, it has the opposite effect when the number of captured nodes increases. This is because the qComp is probabilistic, and to increase the probability of sharing q keys between any two nodes, the size of the key pool must shrink. Thus, capturing a large number of nodes compromises more links than capturing a small number of nodes. However, our scheme is deterministic, and sharing $d-1$ keys between any two nodes with Hamming distance of one from each other is guaranteed by the structure of our scheme.

The final key established between the nodes I and I' is referred to as the link key. This key, denoted by $k_{I,I'}$, is a symmetric function $\chi : \mathbb{F}^{d-1} \rightarrow \mathbb{F}$ of all the $d-1$ common keys, i.e.,

$$k_{I,I'} = \chi\Big(k_{I,I',\ell} : \forall \ell \in [\,d\,] \setminus \{\,j\,\}\Big) \ . \tag{7.22}$$

This function is public to all nodes including the adversary. It should possess the following properties:

1. Its evaluation must be fast and efficient considering the limitations of the senor nodes and
2. It must be symmetric with respect to any ordering of the input arguments for the following reasons:
   a. It enhances the establishment of the link key regardless of the ordering of the participating common keys and
   b. It gives a uniform distribution to the link key when the input arguments vary over all possible values. Therefore, when compromising a link key, all the $d-1$ common keys are equally important. Consequently, an adversary has to obtain all of them to compromise a link key.

It is straightforward to verify that both of the following functions satisfy the aforementioned

properties.

$$\chi(\, x_1, \ldots, x_{d-1}\,) = x_1 \oplus \cdots \oplus x_{d-1} \tag{7.23a}$$

$$\chi(\, x_1, \ldots, x_{d-1}\,) = H(x_1 || \cdots || x_{d-1}) \tag{7.23b}$$

Here, the symbols $\oplus$, $||$, and H denote the bitwise exclusive-OR, the concatenation of two finite-field elements considered as bit strings, and a hash function, respectively. The following example shows how a link key is established.

EXAMPLE 7.3.2. *Let $d = 3$. Consider the two nodes $I = (\, i_0, i_1, i_2\,)$ and $I' = (\, i_0, i_1, i'_2\,)$. Since the Hamming distance between these two nodes is one, they can establish exactly two common keys $k_{I,I',0} = f^0_{i_0}(\, i_1, i_2, i'_2\,) = f^0_{i_0}(\, i_1, i'_2, i_2\,)$ and $k_{I,I',1} = f^1_{i_1}(\, i_0, i_2, i'_2\,) = f^1_{i_1}(\, i_0, i'_2, i_2\,)$. The link key is $k_{I,I'} = \chi(k_{I,I',0}, k_{I,I',1})$.*

As explained before, if the Hamming distance between the nodes I and I' is strictly greater than one, then these nodes are unable to establish a link key. However, there might exist a path connecting these nodes such that every two adjacent nodes on the path are at the Hamming distance of one from each other. In this case, the nodes I and I' can exchange keys using the path since every link on the path is secured by a link key. The final secret key established between I and I' is called *path key*. The establishment of such key is part of some previous KPS such as HB. The disadvantage of establishing a path key is some communication overhead that is imposed on the nodes connecting I and I'. For this reason, we have not included path key in our protocol.

In Section 7.4.1, we obtain the probability of network connectivity. In addition, we show how to choose the protocol parameters such that the network has a giant component. The presence of a giant component implies that there are only very few isolated nodes in the network. We consider a network with this configuration as functional. Hence, the MKPS setting eliminates the excessive communication overhead of the path-key establishment. However, in HB, the network is attempted to be connected. To satisfy full connectivity, one requires to establish path keys. Nevertheless, in MKPS, we may establish a path key only

in a situation where it is critical for an isolated node to establish a secret key with another node with Hamming distance greater than one.

7.4 Evaluation of the MKPS

In this section, we evaluate the proposed scheme. The evaluation metrics are the network connectivity, the probability of the link-key compromise, the communication range, and the storage memory of every node. Another critical metric is the communication overhead. We note that link keys are established without any communication overhead; only path-key establishment introduces communication overhead. Since we try to avoid establishing path keys, as explained in Section 7.3.2, the communication overhead of the MKPS becomes much lower than that in other similar protocols. Throughout this section, we assume that n is the actual number of nodes in the network and $m = \lceil \sqrt[d]{n} \rceil$, where d is the length of the node ID or, in other words, the *dimension*.

7.4.1 Network Connectivity

Similar to previous scheme, we employ the random graph model $\mathcal{G}(n, P_{lk})$ to study the connectivity properties of the network. To calculate the probability of link-key establishment in the proposed MKPS scheme, recall that every two nodes at the Hamming distance of one from each other can establish a link key. Hence, as proved in Appendix D, the average probability of the link-key establishment is

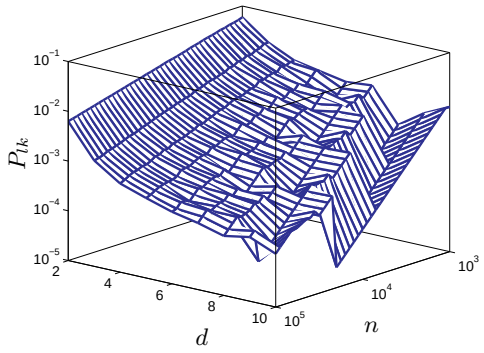
$$\begin{aligned} P_{lk} &\approx \frac{\big[d\,(m-2)+\nu\big](m-1)^d}{n\,(n-1)} + \frac{\theta\, m^d \Big[m\,(d-1)+\nu\,\theta^{\nu-1}+m\,(1-d)\,\theta^{\nu}\Big]}{n\,(n-1)} \\ &\leq \frac{d\,(m-1)}{n-1}\ , \end{aligned} \tag{7.24}$$

where $\theta = 1 - \frac{1}{m}$ and ν is given in the appendix by (D.8).

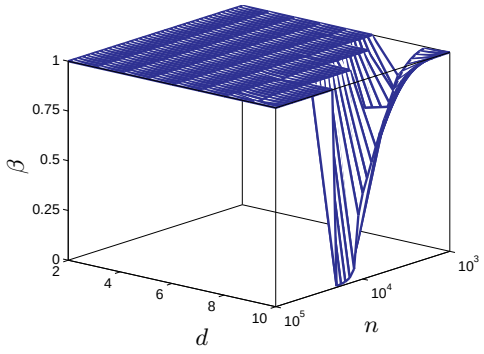
The probability P_{lk} is plotted in Figure 7.2a versus the total number of nodes n and the length d of the nodes IDs. The abrupt jumps in this curve are due to the ceiling function $\lceil \cdot \rceil$ in the definition of m. As the figure shows, for a fixed d, the average probability of the

link-key establishment P_{lk} decreases by increasing the number of nodes n in the network. However, by fixing n and increasing d, the probability P_{lk} is globally decreasing, but it has linearly increasing segments. It globally decreases because m exponentially decreases with d although there is a term linear in d in the first fraction of P_{lk} in (7.24). The linear increasing segments of P_{lk} correspond to the range of n for which m is constant and henceforth, the linear coefficient of d has the dominant effect.

The relation between the local connectivity (the probability of the link-key establishment) and the global connectivity of the whole network can be deduced from the "phase transition" behavior of the random graphs studied by Erdős and Rényi [76, 109]. They



(a) Average probability of the link-key establishment.



(b) Size of the largest component in the network normalized to the network size.

Figure 7.2: Probability of the link-key establishment in the MKPS and its effect on the size of the largest component of the network.

showed that for any monotone property of the random graphs (such as connectivity), there exists a value of P_{lk} for which the property moves from "nonexistent" to "certainly true" in a very large random graph. In order to summarize their result in a theorem, let $c := nP_{lk}$. They discovered that when $c < 1$, $\mathcal{G}(n, P_{lk})$ consists of small components the largest of which has $\Theta(\ln n)$ nodes. By increasing P_{lk} and correspondingly c, particularly when $c > 1$, small components join together to form a *giant component* of size $\Theta(n)$. The other components are relatively small. Erdős and Rényi showed how to compute the local connectivity required for a given global connectivity. The following theorem gives the size of the largest component of the network [109].

THEOREM 7.4.1. *Consider the random graph model $\mathcal{G}(n, P_{lk})$. Let $c = nP_{lk} > 0$ and $0 \leq \beta \leq 1$ be the order of the largest component in the network normalized to the size of the network n. (We note that β is a function of both n and P_{lk}.) The following assertions hold with probability $\to 1$ as $n \to \infty$.*

(i) If $c < 1$, then $\beta \leq \dfrac{3 \log n}{n(1-c)^2}$.

(ii) If $c > 1$, then there is a giant component in the network with size $(1 + o(1))\beta n$, where β is the solution to the equation $\beta + e^{-\beta c} = 1$.

Furthermore, the size of the second largest component is at most $16c \log n/(c-1)^2$.

We note that the presence of the giant component in the network may be sufficient for its healthy operation [106]. Ignoring a few isolated sensor nodes in the network does not considerably degrade the functionality of the network. However, it reduces the required probability of the link-key establishment. As we will discuss in the next subsection, by this tradeoff, we can increase the dimension d that results in the reduction of the probability of the link-key compromise through the dimension-optimization procedure. Moreover, the connectivity of the entire network may be impossible in many practical cases [106]. Therefore, in our scheme, in contrast to the previous schemes, we only guarantee the presence of the giant component in the network that is a more practical assumption. In the proposed MKPS, the normalized size of the largest component β is plotted in Figure 7.2b versus n

and d. As the figure shows, in most of the cases, the largest component is, in fact, the giant component that implies the network is almost connected. However, for small-size networks, the size of the largest component drastically drops when d is increased beyond some level. This decrease in the size of the largest component is due to the abrupt decrease of the probability P_{lk} in Figure 7.2a.

So far, as previous schemes [77, 32, 71, 127], we have used the random graph model $\mathcal{G}(n, P_{lk})$ to represent the network graph resulting from a KPS. In this model, the communication radius of every node is assumed unlimited. However, in practice, every node has only a limited communication range. The relationship between the actual communication radius of the sensor nodes and the probability of the link-key establishment is studied separately in [162] under the network graph model $\mathcal{G}(n, P_{lk}, R)$, where R is the communication range. As proved in [162], the minimum communication radius required to have a connected network is

$$R \propto \sqrt{\frac{\ln n + \zeta}{\pi n P_{lk}}} \ , \tag{7.25}$$

where $\zeta > 0$ is a constant. The probability of link-key establishment P_{lk} only depends on the key distribution graph $\mathcal{G}(n, P_{lk})$. Once P_{lk} is obtained for a KPS scheme, then a minimum communication radius R can be chosen to minimize the power consumption.

In the following subsection, we study the resilience of the network against the node capture when the MKPS scheme is employed.

7.4.2 Resilience Against Node Capture

In the previous subsection, we explained that the network maintains its functionality when most of the nodes are connected together. In this case, the network has a giant component while there might exist a few isolated nodes. An adversary might attack the network by attempting to split the giant component into small components. In order to achieve this goal, the adversary compromises the link keys established between the nodes without physically capturing them. However, by (7.22), a link key between two nodes is a symmetric function of all the $d-1$ common keys established between them. Hence, the adversary has

to compromise all the $d-1$ common keys. We note that by (7.21), the common keys are obtained by evaluating the shares of multivariate polynomials at the nodes IDs. Therefore, without physically capturing the nodes, the adversary has to recover some variables of the multivariate polynomials generating these shares by capturing other nodes in the network that store these shares. The parameters of the scheme determine the minimum number of variables that can be recovered and hence, the minimum number of nodes that must be captured. This observation introduces a threshold effect to the scheme that implies prior to capturing a least number of nodes, the adversary is unable to compromise any link keys. In this subsection, we, first, determine the security threshold of the proposed MKPS scheme. Using this threshold, we calculate the probability of the link-key compromise in the MKPS and compare it to the same probability in other schemes.

Consider two nodes I and I' as in (7.17a) and (7.17b), respectively, with $h\,(I, I') = 1$. By (7.22), the link key $k_{I,I'}$ is a symmetric function of all the $d-1$ common keys $k_{I,I',\ell}$. Hence, the adversary has to compromise all these $d-1$ keys generated by the polynomials $f_{i_\ell}^{\ell}(I\,\langle\ell\rangle\,, x_{d-1})$ and $f_{i_\ell}^{\ell}(I'\,\langle\ell\rangle\,, x_{d-1})$ as in (7.21). However, these polynomials are stored only in the memories of I and I' that are unavailable to the adversary. Thus, for every $\ell \in [\,d\,] \setminus \{\,j\,\}$, the adversary has to recover the polynomial

$$f_\ell\,(\,x_{d-r-1}, \ldots, x_{d-1}\,) = f_{i_\ell}^{\ell}\,(\,\hat{\imath}_0, \ldots, \hat{\imath}_{d-r-2}, x_{d-r-1}, \ldots, x_{d-1}\,) \tag{7.26}$$

from its shares distributed in the network for some integer $1 \le r \le d-1$, where $(\,\hat{\imath}_0, \ldots, \hat{\imath}_{d-2}\,) = I\,\langle\ell\rangle$. The shares of this polynomial are $f_\ell\,(\,\hat{\imath}_{d-r-1}, \ldots, \hat{\imath}_{d-2}, x_{d-1}\,)$, where $\hat{\imath}_{d-r-1}, \ldots, \hat{\imath}_{d-2} \in [\,m\,]$. These shares are accessible to the adversary upon capturing other sensor nodes. There are at most m^r shares of this polynomial available in the network. The minimum number of shares required to recover this polynomial is given by the following lemma.

LEMMA 7.4.1. *To recover the polynomial f_ℓ in (7.26) from its shares, the required number of shares is*

$$\lambda(r,t) = \binom{t+r}{r}, \quad 1 \le r \le d-1\ . \tag{7.27}$$

Proof. We have

$$f_\ell\,(\,x_{d-r-1},\ldots,x_{d-1}\,)=\sum_{i=0}^{t} f_{\ell i}\,(\,x_{d-r-1},\ldots,x_{d-2}\,)\,x_{d-1}^{i}\ ,$$

where each $f_{\ell i}(\,x_{d-r-1},\,\ldots,x_{d-2}\,)$ is an r-variate symmetric polynomial of degree t in each variable that has $\binom{t+r}{r}$ coefficients. Hence, $\lambda(r,t)$ shares is required. ∎

If $m^r < \lambda(r,t)$ for some r, then there are not enough shares of the polynomial in the network to recover r variables. Therefore, it is impossible for an adversary to obtain the polynomial in (7.26) for the given value of r. We note that the inequality $m^r < \lambda(r,t)$ does not imply $m^{r+1} < \lambda(r+1,t)$. In other words, we might have $m^{r+1} \geq \lambda(r+1,t)$ in which case there exist enough shares of f to recover $r+1$ variables. Hence, the security threshold is determined by the number of variables for which there exist enough shares in the network to recover that many variables. This result is summarized in the following corollary.

COROLLARY 7.4.1. *If there exists an integer $1 \leq r \leq d-1$ such that $m^i < \lambda(i,t)$ for all integers $1 \leq i \leq r-1$, but $m^r \geq \lambda(r,t)$, then the MKPS is $(\lambda(r,t)-1)$-secure in the network.*

As this corollary states, the first positive integer r, such that $m^r \geq \lambda(r,t)$, is the number of variables for which there exist enough shares in the network to recover the corresponding polynomial. To facilitate future references to the security parameters, we define the security level of the MKPS as follows.

DEFINITION 7.4.1 (r-Variate Secure). *In the settings of Corollary 7.4.1, we say that the MKPS scheme is* r-variate secure.

In an r-variate secure scheme, the adversary has to capture at least $\lambda(r,t)$ nodes to recover the polynomial f_ℓ in (7.26). Recovering this polynomial enables the adversary to compromise all the $d-1$ common keys, and hence the link key, between the two non-captured nodes I and I'. In other words, the MKPS is perfectly secure up to capturing $\lambda(r,t)-1$ nodes. Hence, our scheme is a threshold KPS. The security threshold r is a global parameter of the scheme, i.e., it is independent of the nodes I and I' since m and $\lambda(r,t)$ are global parameters of the network.

Another security notion that is interesting from the theoretical point of view is *perfect secrecy*. This notion addresses the case when capturing any number of nodes except the two nodes I and I' does not reveal any information about the link key established between these two nodes. In the MKPS scheme, perfect secrecy happens when there are insufficient shares in the network to recover any number of variables. In light of Corollary 7.4.1, the necessary and sufficient condition for perfect secrecy can be stated as in the following corollary.

COROLLARY 7.4.2 (Perfect Secrecy). *The MKPS scheme is perfectly secure if and only if* $m^r < \lambda(r,t)$ *for all* $1 \leq r \leq d-1$.

We note that, in practice, the notion of perfect secrecy is unimportant since a network with a high fraction of its nodes captured by an adversary is naturally considered compromised regardless of the fact that the link between two non-captured nodes is still secure.

The probability of compromising the link key established between any two nodes depends on the security threshold of the scheme. Assume that a fraction of p_{nc} nodes in the network are captured and the MKPS is r-variate secure. By Lemma 7.4.1, the adversary has to obtain at least $\lambda(r,t)$ shares of any polynomial to recover r variables of that polynomial. Since the number of shares of a polynomial is a binomially-distributed random variable, the probability of polynomial recovery is

$$P_{pr} = \sum_{i=\lambda(r,t)}^{m^r} \binom{m^r}{i} p_{nc}^i \left(1-p_{nc}\right)^{m^r-i} \quad . \tag{7.28}$$

A common key established between two arbitrary nodes is compromised only when r variables of the polynomial generating the common key are recovered. In order to compromise a link key, all the $d-1$ common keys must be compromised. This is because, by (7.22), the common keys are obtained by evaluating different polynomials that are randomly and independently generated by the sink. Hence, the probability of the link-key compromise is

$$P_{lkc} = P_{pr}^{d-1} \quad . \tag{7.29}$$

This probability versus the fraction of captured nodes p_{nc} is plotted in Figure 7.3 for different values of d. For a fair comparison, we have fixed the memory usage M of each sensor node to

$M = 50$ elements from $\mathbb{F}$. Hence, the polynomial degree in every variable is $t = \lfloor 50/d - 1 \rfloor$ by (7.20). In all these curves, the total number of nodes n is 100,000. As this figure shows, by increasing d, the probability P_{lkc} generally decreases. However, for some values of d this is untrue. Such an irregular behavior is due to the interplay between different parameters in (7.28), which are all related to d, and the ceiling function $\lceil \cdot \rceil$ in the definition of m. In the following, we present a design criteria to choose an optimal value for d that minimizes the probability of link-key compromise while maintaining the network connectivity.

7.4.3 Dimension Optimization

The common characteristic of all the curves in Figure 7.3 is a sharp increase when p_{nc} increases beyond some value. This behavior can be characterized by a threshold $0 < \tau < 1$ (a typical value is $\tau = 0.1$) and a minimal probability p_τ such that $P_{lkc} \geq \tau$ for all $p_{nc} \geq p_\tau$. We refer to p_τ as the *node-capture tolerance* level beyond which we assume that the network does not operate healthy. Since P_{lkc} depends on the dimension d of the vector space $[\, m \,]^d$, the threshold p_τ is also a function of d. In other words, the node-capture tolerance is defined as

$$p_\tau(d) := \min\Big\{ p_{nc} \;\; : \;\; P_{lkc}\,(p_{nc}, d) \geq \tau, \;\; 0 \leq p_{nc} \leq 1 \Big\} \;\; . \tag{7.30}$$

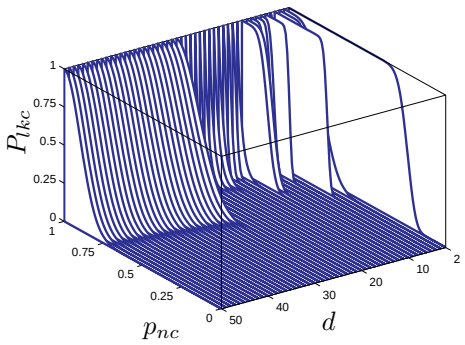


Figure 7.3: Probability of the link-key compromise when $n = 10{,}000$ and $t = \lfloor 50/d - 1 \rfloor$.

To achieve the lowest probability of link-key compromise, we should choose the dimension that maximizes p_τ. However, as shown in Figure 7.2b, by increasing d beyond some value, the size of the largest component of the network, βn, sharply drops. Hence, in the optimization process, we consider only the set of dimensions for which $\beta(d) \geq \beta_0$ for some threshold $0 \leq \beta_0 \leq 1$. (Recall that β is a function of n and P_{lk}, which, in turn, is a function of n and d.) A typical value for this threshold is $\beta_0 = 0.9$ that ensures the graph has a giant component. Mathematically, we define this set as

$$\mathfrak{B}_{\beta_0} := \Big\{ d \in \{\, 2, \ldots, M \,\} \;:\; \beta(d) \geq \beta_0 \Big\} \;, \tag{7.31}$$

where M is the size of the memory used by each sensor node to store the coefficients of the polynomials. The optimal dimension can be considered the one that maximizes $p_\tau(d)$ within $\mathfrak{B}_{\beta_0}$. However, as a rule of thumb in all KPSs, the probabilities of the link-key establishment P_{lk} and the link-key compromise P_{lkc} vary in opposite directions: improving one degrades the other. The probability P_{lkc} directly influences the security of the network. The importance of the probability P_{lk} as a design parameter becomes clear when considering that the transmission radius R of each sensor node is inversely related to the square root of P_{lk}. The transmission power, which is the bottleneck of the energy consumption in sensor networks, is proportional to R^ϵ, where $2 < \epsilon < 4$. Thus, to keep the energy consumption minimal, it is desirable to maximize P_{lk}. However, in general, maximizing P_{lk} increases the probability of the link-key compromise in the network. If prior to the network deployment, we assume that at most a specific fraction of nodes might be captured (i.e., the node-capture tolerance level is known), providing security to the network beyond this level is not an optimal design. Therefore, in contrast to other schemes, we consider both p_τ and P_{lk} as deign metrics. This provides the opportunity to the designer to adjust the network properties according to the desired application. We note that these two metrics are not monotonic functions of the dimension d. (The probability P_{lk} globally decreases with d. However, as shown in Figure 7.2a, there are some local fluctuations that cause the function to be non-monotonic in the mathematical sense.) Therefore, we use the following set in the

optimization process

$$\mathfrak{D}_{\gamma}(p) := \left\{ d \in \mathfrak{B}_{\beta_0} \,:\, p(d) \geq \gamma\, p_{\max} + (1-\gamma)\, p_{\min} \right\} \ , \tag{7.32}$$

where p is either p_{τ} or P_{lk} considered as a function of d. In addition, $p_{\max}$ and $p_{\min}$ are the maximum and the minimum of p over $\mathfrak{B}_{\beta_0}$, respectively, and $0 \leq \gamma \leq 1$ is a design parameter. We note that $\mathfrak{D}_{\gamma}(p)$ is, in fact, the set of all $d \in \mathfrak{B}_{\beta_0}$ for which the value of the function p is above its minimum by $(100\gamma)\%$ of its variation range within $\mathfrak{B}_{\beta_0}$. With this description, for $\gamma = 0$, the set $\mathfrak{D}_{\gamma}(p)$ consists of all $d \in \mathfrak{B}_{\beta_0}$. Similarly, for $\gamma = 1$, this set consists of all $d \in \mathfrak{B}_{\beta_0}$ for which the value of the function p is equal to its maximum within $\mathfrak{B}_{\beta_0}$. These two observations correspond to the two extreme values of the parameter γ that can be mathematically formulated as

$$\mathfrak{D}_0(p) = \mathfrak{B}_{\beta_0}, \qquad \mathfrak{D}_1(p) = \underset{d \in \mathfrak{B}_{\beta_0}}{\operatorname{argmax}}\, p(d) \ . \tag{7.33}$$

Therefore, the optimal dimension is

$$d_{\mathsf{opt}} := \max\left(\mathfrak{D}_{\gamma_1}\left(p_{\tau}\right) \cap \mathfrak{D}_{\gamma_2}\left(P_{lk}\right)\right) \ , \tag{7.34}$$

where $0 \leq \gamma_1, \gamma_2 \leq 1$ are free design parameters. By the two observations in (7.33), the parameters γ_1 and γ_2 in (7.34) determine the importance of lowering the probability of the link-key compromise (as a security metric) and increasing the probability of the link-key establishment (as a network-connectivity metric) in the dimension-optimization process, respectively. As explained before, these two design metrics vary in the opposite directions. Hence, choosing the values of γ_1 and γ_2 both close to 1 results in disjoint sets $\mathfrak{D}_{\gamma_1}\left(p_{\tau}\right)$ and $\mathfrak{D}_{\gamma_2}\left(P_{lk}\right)$. As a rule of thumb, by increasing one of these two parameters, the other one must be decreased. The two extreme cases are:

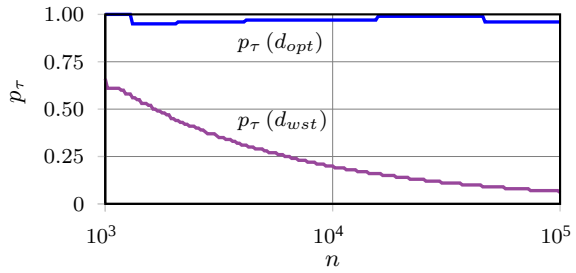
- $\gamma_1 = 1$ and $\gamma_2 = 0$: In light of (7.33), this choice for the parameters maximizes the node-capture tolerance p_{τ} without considering P_{lk} in the optimization.
- $\gamma_1 = 0$ and $\gamma_2 = 1$: Similarly, this setting maximizes only P_{lk} (hence, minimizes the power consumption) without considering the security parameter p_{τ}.

We note that $\mathfrak{D}_{\gamma_1}(p_\tau) \cap \mathfrak{D}_{\gamma_2}(P_{lk})$ is the set of dimensions in $\mathfrak{B}_{\beta_0}$ that satisfy the desired security and connectivity properties. We take the maximum dimension in this set as the optimal one in favor of the security. Similar to (7.34), the worst dimension d_{wst} is defined as

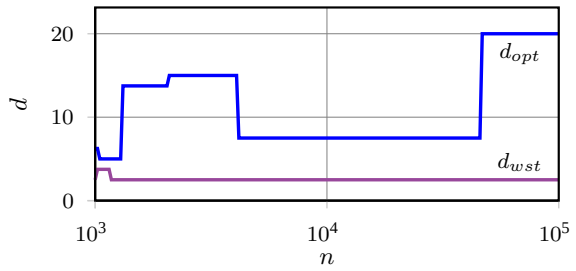
$$d_{\mathsf{wst}} := \min\big(\mathfrak{D}^c_{1-\gamma_1}(p_\tau) \cap \mathfrak{D}^c_{1-\gamma_2}(P_{lk})\big) \ , \tag{7.35}$$

where the superscript c denotes the set complement. The curves of d_{opt} and d_{wst} versus the network size n are plotted in Figure 7.4a for $\gamma_1 = 1$ and $\gamma_2 = 0$. To show the importance of choosing the optimal dimension, the curves of $p_\tau(d_{opt})$ and $p_\tau(d_{wst})$ versus the network size are depicted in Figure 7.4b. As this figure shows, the difference between these values reaches as high as 0.93, which is very significant.

The probabilities of the link-key compromise in the EG of [77], the qComp of [32], the HB of [127], and the proposed MKPS are compared to each other in Figure 7.5. For a fair comparison, in plotting all these curves, the node memory M and the probability of

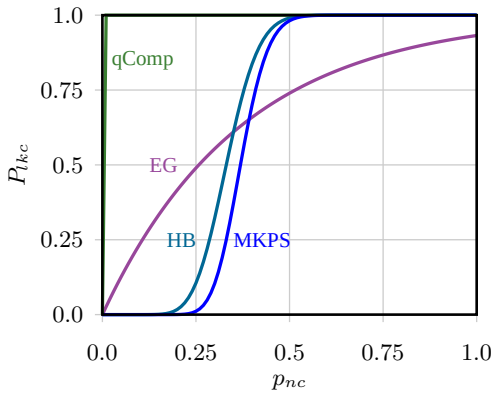


(a) The optimal dimension d_{opt} and the worst dimension d_{wst}.

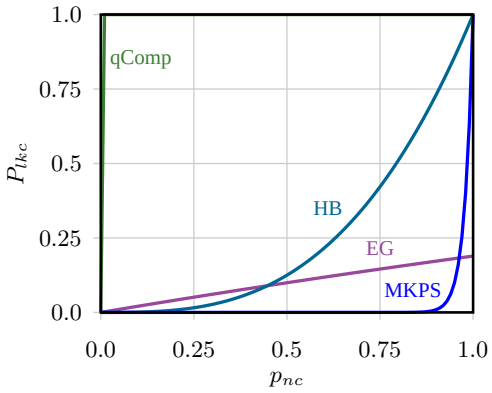


(b) Curves of $p_\tau(d_{opt})$ and $p_\tau(d_{wst})$.

Figure 7.4: Optimal versus the worst dimension and the corresponding p_τ. The memory size is $M = 50$ and the thresholds are $\tau = 0.1, \beta_0 = 0.9, \gamma_1 = 1$, and $\gamma_2 = 0$.



(a) In these curves, the optimal dimension in the MKPS is $d_{\text{opt}} = 16$ that is obtained by choosing $\gamma_1 = 1$ and $\gamma_2 = 0$. The corresponding probability of link-key establishment is $P_{lk} \approx 10^{-4}$. The sizes of the key pool in the EG and qComp are $P = 23{,}840{,}419$ and $P = 3{,}408$, respectively.



(b) In these curves, the optimal dimension in the MKPS is $d_{opt} = 3$ that is obtained by choosing $\gamma_1 = 0.01$ and $\gamma_2 = 0.2$. The corresponding probability of link-key establishment is $P_{lk} \approx 10^{-3}$. The sizes of the key pool in the EG and qComp are $P = 1{,}858{,}772$ and $P = 3{,}354$, respectively.

Figure 7.5: Probability of the link-key compromise versus the fraction of captured nodes for different schemes. In all these curves, $n = 100{,}000$, $M = 50$, $\tau = 0.1$, $\beta_0 = 0.9$, and $q = 3$ in the qComp.

the link-key establishment P_{lk} are fixed. In Figure 7.5a the emphasis is on the security metric P_{lkc} by choosing $\gamma_1 = 1$ and $\gamma_2 = 0$. In contrast, in Figure 7.5b, the values of these parameters are $\gamma_1 = 0.01$ and $\gamma_2 = 0.2$ for which the resulting KPS provides a high network connectivity. As these curves show, the MKPS achieves the lowest probability of the link-key compromise with the same memory size and network connectivity probability. For example, in Figure 7.5a, when 90% of the nodes are captured, the fraction of compromised links among non-captured nodes in the MKPS scheme is 0.8%. However, the same fraction in the HB, EG, and qComp is 73%, 17%, and 100%, respectively. We note that the probability P_{lk} in Figure 7.5b is 10 times that in Figure 7.5a. As discussed before, this implies a reduction in the transmission radius by a factor of $10^{1/\epsilon}$, where $2 \leq \epsilon \leq 4$.

7.4.4 Communication Range

As explained before, communication transmission and reception are the most energy consuming operations in a sensor node. To prolong the network lifetime, the communication radius should be minimized. The minimum communication radius, in any KPS, required to have a connected network is given in (7.25). To incorporate the effect of captured nodes in the network connectivity, we use the actual probability of the link-key establishment P_{lkc}^{act} in (7.25). This probability is related to P_{lk} as follows

$$P_{lk}^{act} = P_{lk}\ (1 - P_{lkc}) \quad , \tag{7.36}$$

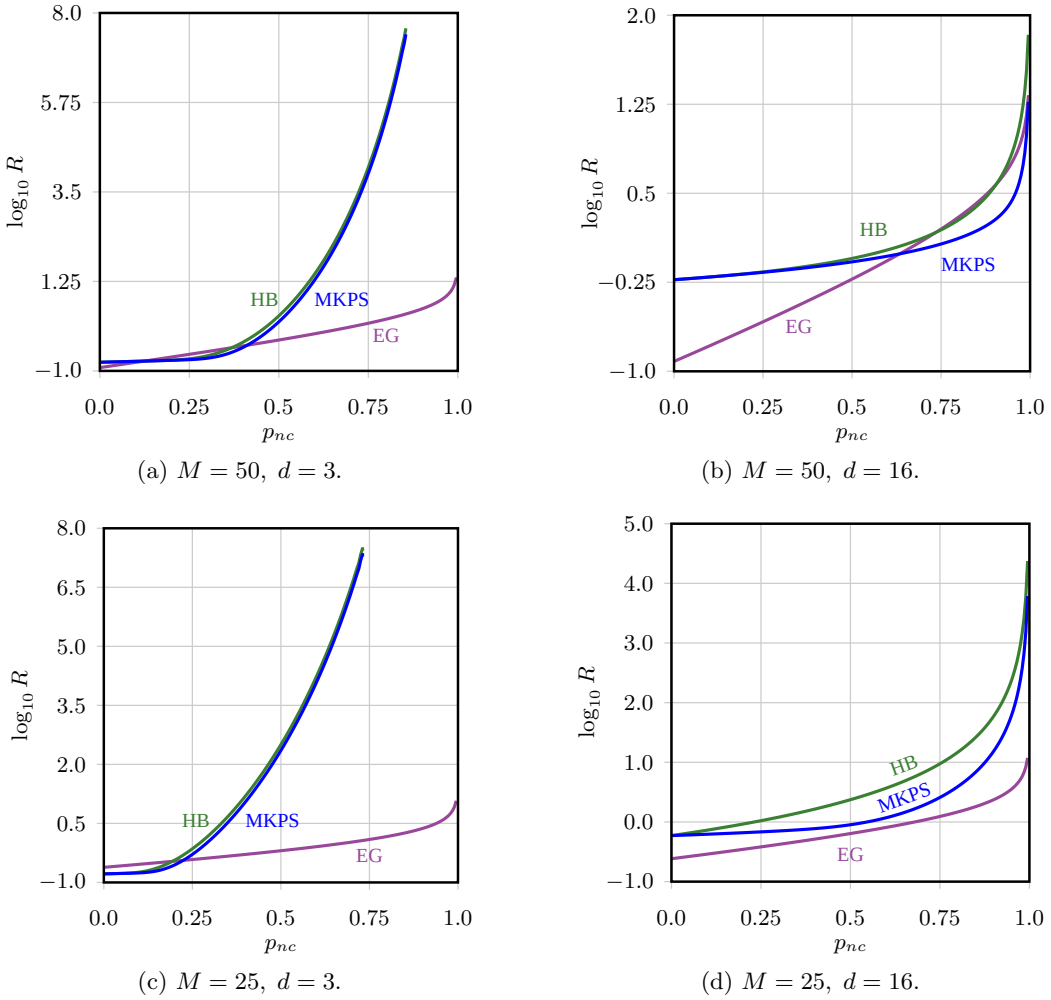
which reflects the fact that a link between two nonmalicious nodes is considered healthy if:

1. these nodes have been able to establish a link key after the network deployment and before any adversarial activities and
2. the link established between these nodes is not compromised.

We have compared communication radii of the EG, HB, and MKPS in Figure 7.6[1]. In all these curves, the size of the network is n = 100,000 and the size of the key pool in the EG scheme is P = 1,000,000. Based on these curves, the proposed MKPS requires a communication radius less than the HB does. The difference becomes significant for large values of the dimension d. This observation supports the fact that while maintaining the same network connectivity, the MKPS provides network resiliency much higher than the HB does. For example, for $M = 50$ and $d = 3$, we have $R_{\text{EG}} : R_{\text{HB}} : R_{\text{MKPS}} = 1 : 4.0466 : 2.8681$ when 50% of the nodes in the network are captured. By increasing the dimension to $d = 3$, this ratio becomes $R_{\text{EG}} : R_{\text{HB}} : R_{\text{MKPS}} = 1 : 1.4946 : 1.3981$.

Another observation in Figure 7.6 is that for small sizes of the node memory, the effect of increasing the dimension becomes more significant. For example, we have $R_{\text{EG}} : R_{\text{HB}} : R_{\text{MKPS}} = 1 : 1.8747 : 1.3497$ for $M = 25$, $d = 3$, and $p_{nc} = 0.25$. The same ratio for $d = 16$ is $R_{\text{EG}} : R_{\text{HB}} : R_{\text{MKPS}} = 1 : 2.7496 : 1.7861$.

[1] In plotting these curves, we have ignored the constant coefficient of the communication radius in (7.25).



(a) $M = 50,\ d = 3.$ (b) $M = 50,\ d = 16.$

(c) $M = 25,\ d = 3.$ (d) $M = 25,\ d = 16.$

Figure 7.6: Communication radius versus the fraction of captured nodes for different schemes. In all these curves, $n = 100{,}000$ and the size of the key pool in the EG scheme is $P = 1{,}000{,}000$.

7.4.5 Storage Memory

Every sensor node I requires two different types of storage memory:

1. One type with amount M_k bits to store its ID and the coefficients of the d univariate polynomials each of degree t with coefficients from $\mathbb{F}$.

2. Another type with amount M_c to store the IDs of captured nodes that are at the the Hamming distance of one from it. This is because I is able to establish link keys with these nodes. If they are captured, all communications with them must be abandoned.

As explained before

$$M_k = M \log_2 |\mathbb{F}| + d \log_2 m \ , \tag{7.37}$$

where M is the maximum capacity of every sensor node, in terms of the number of elements in $\mathbb{F}$, to store the coefficients of the polynomials.

To determine M_c for an arbitrary sensor node I, assume that the MKPS is r-variate secure as in Definition 7.4.1. Let f be a multivariate polynomial such that I stores one of its shares. The r-variate secrecy assumption implies that without capturing I, an adversary has to capture at least $\lambda(r,t)$ other nodes that also store the shares of f in order to recover r variables of f. Let $\mathcal{N}_f$ be the set of all sensor nodes in the network that store the shares of the multivariate polynomial f. The node I can establish a link key with another node $I' \in \mathcal{N}_f$ if and only if $h(I, I') = 1$. We note that there are at most $m-1$ such nodes in the network. If $\lambda(r,t) \leq m-1$, then I has to store the IDs of at most $\lambda(r,t)-1$ such nodes when they are captured by the adversary. The rationale is that if more nodes are captured, then f is compromised, and I can delete the share of f from its memory. This case happens when the scheme is univariate secure, i.e., when $r=1$. Since $\lambda(1,t) = t+1$, we have $M_c \leq dt \log_2 m$. (Recall that if $h(I, I') = 1$, then the IDs of I and I' differ in only one coordinate that may be stored by I.) Considering the maximum value of M_k, we conclude that when the scheme is univariate secure, we have

$$M_k + M_c \leq d(t+1)\big(\log_2 |\mathbb{F}| + \log_2 m\big) \ . \tag{7.38}$$

The required node storage-memory in the HB of [127] is the same.

The other possible case is $\lambda(r,t) \geq m$. In this case, the adversary cannot compromise the polynomial f by capturing all the $m-1$ nodes $I' \in \mathcal{N}_f$ such that $h(I, I') = 1$. However, if the adversary does so, the node I cannot use the share of f to establish a link key with another node in the network. Hence, the node I can delete the share of f from its memory although the polynomial may not be compromised. We conclude that in this case, the node I has to store the IDs of at most $m-2$ other nodes $I' \in \mathcal{N}_f$ such that $h(I, I') = 1$. Hence,

$M_c \leq d(m-2)\log_2 m$. Using the maximum value of M_p, we conclude

$$M_k + M_c \leq d(t+1)\log_2 |\mathbb{F}| + d(m-1)\log_2 m \ . \tag{7.39}$$

To compare the memory usage in this case with that in the HB, we note that $\lambda(r,t) \geq m$ when $r > 1$. Moreover, this implies $m \leq t$ by the definition of r-variate secrecy. Thus, comparing (7.39) with (7.38), we conclude that the memory usage in this case is lower than that in the previous case. In addition, we note that the memory usage in the HB is the same as that when we have univariate secrecy. Therefore, the final conclusion is that the node memory-usage in the proposed MKPS is less than or equal to that in the HB.

7.5 Location-Aware MKPS

A sensor network may randomly be deployed in a field in which case there is no prior knowledge as to which sensors are located at the vicinity of each other. The MKPS is an example of such schemes. In static sensor networks, although it is difficult to precisely determine the location of the sensor nodes in the field, some approximate information are sometimes available. For example, when a truck is used to deploy a network of static sensors, we may arrange them on the points of a two-dimensional lattice prior to the deployment. After the deployment, the sensors may not position on their preset locations, but their approximate locations are known. We may take advantage of this information to design a KPS that provides higher network connectivity comparing to the schemes that use the random-deployment model.

In this section, we propose a location-aware multivariate key pre-distribution scheme (LA-MKPS) based on the proposed MKPS. In this new scheme, the deployment field is divided into non-overlapping hexagonal cells of equal areas. The sensors are uniformly distributed among the cells by partitioning them into groups of almost equal size. The LA-MKPS consists of two key pre-distribution layers. One layer is used for the secure communication within a cell for which we use the MKPS. The second layer is designed

for communications between adjacent cells. For this layer, we adopt an MKPS based on bivariate polynomials. The details of the LA-MKPS are provided in this section.

The first step in the LA-MKPS is partitioning the deployment field into non-overlapping cells of equal areas. For this purpose, we consider cells of hexagonal shape based on the observation that sensors usually employ omnidirectional antennas [159]. Hence, similar to mobile communication systems, a honeycomb-like structure of communication cells provides the most efficient coverage [181, 110]. The advantage of using hexagons over squares is that the deployment field can be covered with smaller number of cells with the former choice. The coverage of the hexagonal cells versus that of the square cells is compared in Figure 7.7. As this figure shows, a hexagonal cell gives a better approximation of the circular wireless transmission-coverage of a sensor. Hence, a hexagonal cell with the maximum lateral dimension of $2R$ covers a larger area than a square cell with the same lateral dimension. The corresponding areas of these cells are $A_{sq} = \frac{(2R)^2}{2}$ and $A_{hex} = \frac{3\sqrt{3}(2R)^2}{8}$ with the ratio $A_{sq}/A_{hex} = 1.3$.

Assuming that the wireless communication range of the sensors is R, we cover the target field by non-overlapping hexagonal cells with lateral dimension $2R$. If the area of the field is A, the minimum number of cells required to cover the field is

$$C = \left\lceil \frac{8\sqrt{3}}{9} \frac{A}{R^2} \right\rceil \quad . \tag{7.40}$$

This choice guarantees that all sensors in a cell are in the communication range of each other. The nodes are uniformly assigned to the cells. Thus, all the cells have almost equal number of nodes. To secure the inter-cell communications, we employ MKPS by

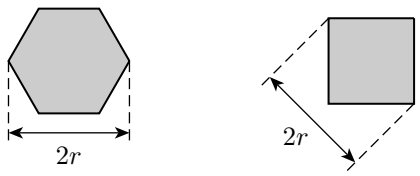


Figure 7.7: Square versus hexagonal cell with the same communication range.

distributing the shares of multivariate polynomials to all the sensors in a cell. Since these sensors are all in the communication range of each other, every two of them can establish a link key. To localize the effects of the link compromise, we use a different pool of multivariate polynomials for every cell. The MKPS employed in every cell can be designed to provide perfect secrecy as in Corollary 7.4.2.

To facilitate future references, frequently used notations are listed below with their meanings.

C	Number of hexagonal cells
M_k^c	Size of the memory to store coefficients of the bivariate polynomial shares
M_c^c	Size of the memory to store IDs of compromised groups in adjacent cells
m_c	$= \lceil \sqrt{C} \rceil$
t_c	Degree of bivariate polynomials in both variables
p_c	Order of the finite field from which the coefficients of the bivariate polynomials are taken
G	Number of groups in every cell
$(i,j)_g$	g-th group in cell (i,j)
n_g	Number of sensors in every group in every cell
n_c	Number of sensors in every cell
P_{lkc}^c	Probability of the link-key compromise
p_{psc}	Probability of compromising a share of a bivariate polynomial

7.5.1 Setup

To distribute keys required for the secure communication between adjacent cells, we use a grid-based approach. In this approach, we assign the points on a two-dimensional grid to the cells. In addition, we assign a unique symmetric bivariate polynomial to every cell. To distribute the shares of this polynomial between sensors, we divide the sensors in every

cell into equal-size groups. In every cell, the shares of the corresponding polynomial are distributed among the sensors in the groups. Moreover, the sensors of a cell store the shares of the polynomials corresponding to the neighbor cells. As a result, the neighbor cells are able to establish pairwise keys. The setup algorithm is explained in an algorithmic language in the following.

1. If C is the total number of cells, design a two-dimensional grid with size $m_c \times m_c$, where $m_c := \lceil \sqrt{C} \rceil$. To every cell, assign a unique point (i, j) on the grid, where $i, j \in (\, m_c \,)$.

2. Design a pool of m_c^2 symmetric bivariate polynomials $f_{i,j}(x, y) \in \mathbb{F}_{p_c}[x, y]$ with degree t_c in both variables. For all $i, j \in (\, m_c \,)$, assign the polynomial $f_{i,j}(x, y)$ to the cell (i, j).

3. Divide the sensors in the cell (i, j) into G almost-equal-size disjoint groups labeled by $(i, j)_0, \ldots, (i, j)_{G-1}$. Let n_g be the maximum number of sensors in every group. For any $g \in (\, G \,)$, store the coefficients of the polynomial $f_{i,j}(g, y)$ in all the sensors in $(i, j)_g$.

4. As shown in Figure 7.8, assume the six neighbors of the cell (i, j) are (i_ℓ, j_ℓ) for $\ell \in (\, 6 \,)$. Store the coefficients of the six polynomials $f_{i_\ell, j_\ell}(g, y)$ to all the sensors in $(i, j)_g$ for all $g \in (\, G \,)$.

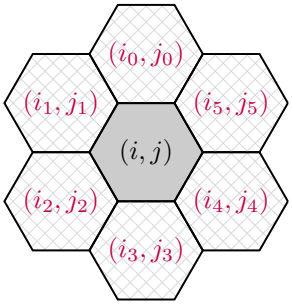


Figure 7.8: A hexagonal cell and its six neighbors.

Using this scheme, every sensor stores $7(t_c+1)\log_2 p_c$ bits in addition to $d(t+1)\log_2 p$ bits for the MKPS employed in every cell.

7.5.2 Link-Key Establishment

Every two nodes in two adjacent cells that are in the communication range of each other can establish a link key using the proposed LA-MKPS. Consider two sensors I and I' respectively belonging to two groups $(i,j)_g$ and $(i',j')_{g'}$ in adjacent cells (i,j) and (i',j') for arbitrary $g, g' \in [G]$. These nodes both store the shares of the same bivariate polynomials as follows.

$$I \ : \ f_{i,j}(g,y),\ f_{i',j'}(g,y)$$
$$I' \ : \ f_{i',j'}(g',y),\ f_{i,j}(g',y)$$

Using these shares, the nodes I and I' calculate two common keys as:

$$k_{I,I',0} = f_{i,j}(g,g') = f_{i,j}(g',g) \tag{7.41a}$$
$$k_{I,I',1} = f_{i',j'}(g',g) = f_{i',j'}(g,g') \ . \tag{7.41b}$$

The link key $k_{I,I'}$ between these nodes is obtained by the application of a function χ as in (7.23) with two inputs, i.e.,

$$k_{I,I'} = \chi\left(k_{I,I',0}, k_{I,I',1}\right) \ . \tag{7.42}$$

7.6 Evaluation of the LA-MKPS

Every two nodes within a cell are in the communication range of each other by our design. Hence, by the properties of the MKPS, every two such nodes can establish either a link or a path key. Moreover, as explained in the previous section, every two nodes in adjacent cells, which are in the communication range of each other, can establish a link key. Therefore, the LA-MKPS provides complete connectivity to neighbor sensor nodes. This idealistic feature is obtained by taking advantage of the location information. In the remaining of

this section, we study the resiliency of the LA-MKPS against the node capture and also its required storage memory.

7.6.1 Resiliency Against the Node Capture

As explained before, the LA-MKPS consists of two KPS layers: one for the inner and one for intra-cell communications. Since we use the proposed MKPS for the inner-cell communications, the resiliency analysis of this layer against the node capture is provided in Section 7.4.2. Hence, here, we study the resiliency of the intra-cell communications against the node capture.

There are G shares of every symmetric polynomial $f_{i,j}(x, y)$ available in the network that are $f_{i,j}(\ell, y)$ for all $\ell \in (G)$. Since all these polynomials are bivariate, one of the following two cases is possible.

Perfect Secrecy: In this case, there are not enough shares of any bivariate polynomial in the network to recover that polynomial. Since the degree of all bivariate polynomials in both variables is t_c, we must have $G \leq t_c$. Although perfect secrecy is an attractive feature, we note that all the sensor nodes in a group use the same shares of the same bivariate polynomials. Hence, capturing only one node in a group, compromises all the communications within that group. This observation suggests increasing the number of groups although it results in loosing perfect secrecy.

Univariate Secrecy: This case arises when $G > t_c$. With an analysis similar to that performed in Section 7.4.2, we deduce that the probability of the link-key compromise is

$$P_{lkc}^{c} = \left[1 - \sum_{i=0}^{\min(t_c,G)} \binom{G}{i} p_{psc}^{i} \left(1 - p_{psc}\right)^{G-i}\right]^{2} \quad , \tag{7.43}$$

where p_{psc} is the probability of compromising a polynomial share. Since a polynomial share, which is distributed among all the nodes of a group, is compromised if and only if at least one of the nodes in that group is compromised, we have

$$p_{psc} = 1 - (1 - p_{nc})^{n_g} \quad , \tag{7.44}$$

where p_{nc} is the probability of the node capture and n_g is the number of nodes in every group. If there are n_c sensors in every cell, then $n_g = \lceil \frac{n_c}{G} \rceil$.

In Figure 7.9, we compare P_{lkc}^c in the LA-GB of [126] and the proposed LA-MKPS. In these curves, the assumption is that there are $n_c = 100$ sensors in every cell. We note that in the LA-GB, every sensor stores five polynomials while in the LA-MKPS, seven polynomials are stored in every sensor. To take into account this difference, we have adjusted the value of t_c in our comparison as $t_c = \lfloor 75/d - 1 \rfloor$, where $d = 5$ in the LA-GB and $d = 7$ in the LA-MKPS. As these curves show, the LA-MKPS has a lower probability of the link-key compromise. For example, when 20% of the sensors are captured, about 92% of the link keys in the LA-GB are compromised. However, in the LA-MKPS, only 10% of the link-keys are compromised. Another observation is that by increasing the number of sensors n_g in each group, the probability P_{lkc}^c further decreases. This is due to the inverse relationship between n_g and G that affects P_{lkc}^c in (7.43).

7.6.2 Storage Memory

In this section, we calculate the amount of the storage memory required by every sensor node to store the keying materials. Since for the first layer we use MKPS, the amount of the storage memory for this layer is given in Section 7.4.5. Therefore, we only focus on the amount of storage memory for the second layer. Similar to the MKPS, every node requires

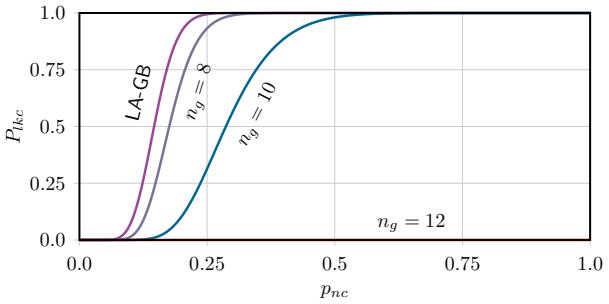


Figure 7.9: Probability of the link-key compromise versus the fraction of captured nodes. Parameters are $n_c = 100$, $t_c = 14$ in the LA-GB, and $t_c = 9$ in the LA-MKPS. The memory usage of each sensor is 75.

two different types of storage memory:

1. The first type is to store the coefficients of the seven polynomial shares, the ID the cell in which this node is residing, and the IDs of the six adjacent cells. If M_k^c is the amount of this memory in bits, then

$$M_k^c = 7\,(t_c + 1)\log_2 p_c + 12\log_2 m_c \ . \tag{7.45}$$

2. The second type of memory is required to store the IDs of the groups in adjacent cells for which at least one of the belonging nodes is captured. We denote the amount of this memory by M_c^c in terms of the number of bits.

To determine M_c^c, consider an arbitrary node I in the cell $\mathfrak{C} = (i, j) \in (\, m_c\,)^2$. Let $\mathfrak{C}' = (i', j')$ be a cell adjacent to $\mathfrak{C}$. If for some $\ell \in (\, G\,)$, a node $I' \in \mathfrak{C}'_\ell$ is captured, then I must abandon all communications with every node in the group $\mathfrak{C}'_\ell$. By (7.42), every link key established between I and a node $I' \in \mathfrak{C}'$ requires a share of the polynomial $f_{i',j'}(x, y)$. If this polynomial is compromised, then I cannot establish a link key with any node in $\mathfrak{C}'$. Thus, the number of groups in $\mathfrak{C}'$ that the node I has to store their IDs in case they are compromised depends on the security level of the second layer. First, assume the second layer is perfectly secure, i.e., $G \leq t_c$. Since the cell $\mathfrak{C}$ has six adjacent cells, we have $M_c^c \leq \log_2 6 + \log_2 g$. If the second layer is univariate secure, then the node I has to store the IDs of at most t_c groups in every cell. Therefore, we have

$$M_c^c \leq \log_2 6 + \log_2 \min\,(G, t_c) \ . \tag{7.46}$$

In LA-GB, we have $M_k^c = 5\,(t_c + 1)\log_2 p_c + 2\log_2 m_c$ [126]. Moreover, $M_c^c \leq 2 + \log_2 \min\,(n_c, t_c)$, where n_c is the total number of nodes in every cell. To compare the total amount of storage memory between the proposed LA-MKPS and the LA-GB, we provide numerical values for a typical case in the following example.

EXAMPLE 7.6.1. *Consider a field covered by $C = 100$ hexagonal cells each containing $n_c = 100$ sensor nodes. Let $t_c = 9$ and $G = 8$. In the LA-MKPS, we have $M_k^c + M_c^c = 285$ bits. However, in the LA-GB, we have $M_k^c + M_c^c = 205$ bits.*

As the above example reveals, in most cases, the LA-MKPS required slightly larger storage memory comparing to the LA-GB. However, LA-MKPS provides a higher resiliency against the node capture.

7.7 Summary

We proposed two threshold key pre-distribution schemes in this chapter namely MKPS and LA-MKPS. In the MKPS, every sensor node is assigned a unique d tuple of positive integer as its ID. Using these IDs, the shares of multivariate polynomials are pre-distributed among the sensor nodes. After the deployment, some nodes are able to directly establish exactly $d-1$ common keys using their stored shares of polynomials. The secret key between these nodes is a combination of all these $d-1$ keys. Hence, the proposed scheme is, in a sense, a $(d-1)$-composite method. This feature considerably improves the security of the MKPS. Fortunately, this feature is obtained for free with no payoffs such as additional memory. The proposed MKPS has the threshold property, i.e., it remains perfectly secure up to the capture of a certain fraction of sensor nodes. We also proposed an algorithm to find the optimal dimension d in our scheme. In contrast to previous schemes, we provide both security and network connectivity as the optimization criteria. Therefore, the proposed scheme provides an opportunity to the designer to adjust the network properties according to the desired application.

The proposed LA-MKPS scheme takes advantage of the location information to improve the connectivity of the network by partitioning the deployment field into hexagonal cells. The evaluation of this technique shows significant improvement over the previous location-aware schemes.

CHAPTER 8

Data Authenticity and Availability

8.1 Introduction

In this chapter, we consider the data authentication and the data availability in WSNs[1].

Triggered by an event in the field or upon a sink query, the nodes close to the center of stimulus collaboratively generate a report and send it back to the sink. Considering the wide scattering of the nodes in the field, the center of stimulus is usually distanced from the sink rendering single-hop communication with the sink impossible. Therefore, the generated report is forwarded to the sink through multi-hops.

Security of multi-hop data transfer in wireless sensor networks becomes very important especially for the networks deployed in hostile environments. Constraints of sensor nodes and the lack of infrastructure in such networks poses new challenges in designing security services. The major attacks that this chapter is concerned on a wireless sensor network are as follows:

Eavesdropping: By listening to the radio channel, the adversary tries to obtain meaningful information (e.g., traffic monitoring).

False Data Injection: In this attack, an insider node attempts to cause false alarms or to consume the energy of the forwarding sensors by injecting false data.

Data Drop: An insider node drops a legitimate report on the forwarding path toward the sink.

[1]This part of the thesis is developed together with my colleague Mr. Erman Ayday.

Noise Injection: Legitimate reports are modified by injecting noise. Thus, the sink is unable to regenerate the original message.

Cryptographic services required to prevent these attacks are *data confidentiality*, *data authenticity*, and *data availability*.

In this chapter, we propose a new scheme that provides all the aforementioned security services with moderate communication and computation overhead [52, 12]. The proposed scheme makes extensive use of the node collaboration and data redundancy to provide data authenticity and availability. To achieve this goal, we assume that the node scattering is dense enough such that a single event in the field is sensed by more than one sensor node and a message broadcast is received by multiple nodes in the proximity. Every step of the proposed scheme is carried out by multiple nodes involved in the protocol, and all of them generate the same output. Hence, a few malicious nodes can be detected, and the bogus packets generated by them are dropped.

To evenly distribute the load of report generation and forwarding and also enhance the node collaboration, we partition the terrain into non-overlapping cells of the same shape and area. A report generated at the event cell is forwarded toward the sink on the shortest path in a cell-by-cell fashion. The advantages of this technique are localizing adversarial activities and providing a robust and simple routing and authentication mechanism. To provide an authentication mechanism, all the nodes involved in the protocol in every cell generate a hash tree of the same packets. Every node broadcasts only a few packets along with the corresponding authentication information. The nodes in the next forwarding cell check the authenticity of all the received packets and drop bogus ones.

Linear random network coding is an essential component of the proposed scheme [124]. In this type of coding, intermediate nodes process the data by generating random linear combinations of the packets they receive. This technique is advantageous in the erasure channel model since the redundancy in the data allows the sink to recover the original message packets by receiving few encoded packets. The erasure channel also models the packet-drop attack by an adversary. Therefore, random network coding intrinsically pro-

vides a countermeasure to data drop.

8.1.1 Related Work

One of the first works in data authentication for wireless sensor networks is IHA [208]. In this scheme, the sensor nodes are organized into clusters. A legitimate report is generated by the collaboration of a minimum number of nodes inside a cluster. Every cluster has a representative that is called the CH. The CH is responsible for collecting enough number of MACs generated by the collaborating nodes, generating a report, and forwarding it to the sink. At the initialization phase, the sink discovers the forwarding path from every CH to itself by sending a HELLO message down the network and collecting the reflections. These paths are periodically updated to take into account changes in the network topology over time. Using the information gathered about the network structure, nodes connecting every CH to the sink establish *key chains*. To explain in more detail, consider a typical scenario shown in Figure 8.1. All nodes at the same number of hops away from each other form a key chain (also called associated nodes). In Figure 8.1, this number is four hops. For example, the sequences of nodes (v_1, u_1, u_5, u_9), (v_2, u_2, u_6), and (CH, u_4, u_8) are key chains. Every two consecutive nodes on a key chain establish a secret pairwise key that is used for report authentication.

Triggered by an event in the field or queried by the sink, the CH starts compiling a report. For this purpose, CH broadcasts its own sensor reading to all nodes in it cluster. Every cluster node receiving such a message, compares the broadcast reading with its own.

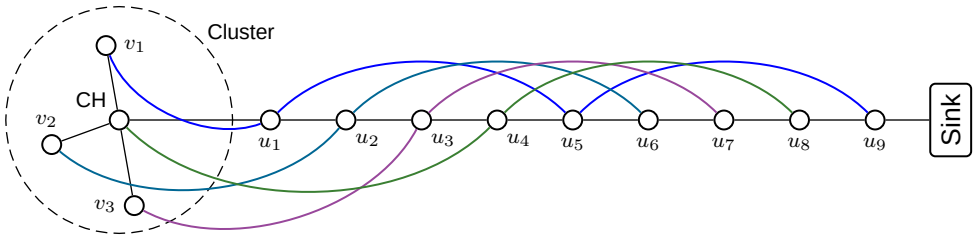


Figure 8.1: Key chains in IHA.

If the difference is within an acceptable range, the cluster node calculates a MAC on the reading of the CH using the secret key common with the next node on the corresponding key chain. The CH collects all MACs and compiles a report as soon as the number of MACs reaches a minimum number. In every hop on the path toward the sink, only one of the many MACs attached to the report is verified. The verified MAC is updated by the enroute node with the secret key common with the next node on the corresponding key chain. The report is dropped by any enroute node if the MAC is unverified. Therefore, a malicious node injecting noise to the network always causes these messages to be dropped. The other drawback of IHA is the key chain maintenance that introduces high communication overhead.

Another approach to data authentication is the SEF proposed in [206]. This scheme is very similar to IHA. The main difference is that associated nodes are not manually determined at the initialization phase. In contrast to IHA, they are discovered by a probabilistic approach. In SEF, every node is pre-distributed with the keying material that are used to establish authentication keys after the network deployment. Key pre-distribution parameters are selected to guarantee, with a high probability, that any CH is able to establish many authentication keys. The SEF provides data availability similar to IHA. Because of the probabilistic nature of SEF, every node is required to store many keys to guarantee the existence of a minimum number of authentication keys. Therefore, two other drawbacks of SEF are the requirement for large storage memory and the possibility of revealing many authentication keys by compromising only a few nodes.

Both previous schemes have a threshold property, i.e., an adversary has to compromise a minimum number of authentication keys to forge a report. To achieve graceful performance degradation to an increasing number of compromised keys, the location-binding keys and location-based key assignment are employed in [205]. The proposed scheme, called LBRS, is conceptually very similar to the SEF. However, the data is forwarded toward the sink in a hop-by-hop fashion. Thus, LBRS localizes the adversarial activities to only the area of the network which is under attack. The LBRS inherits the disadvantages of the SEF except

the performance degradation behavior.

One of the most recent authentication schemes is the LEDS [166]. This is a location-aware scheme that provides many security services such as data confidentiality, availability, and authenticity. In LEDS, the data confidentiality is achieved by using symmetric cryptography and linear secret sharing. To check the authenticity of the data, a legitimate report carries many MACs that are verified by the nodes in the intermediate cells. For the data availability, the overhearing nodes in every forwarding cell collaborate to inform the next cell in case a legitimate report is dropped by a malicious node. Although overhearing nodes theoretically provide data availability, there does not seem to exist a practical method to implement this technique. The most logical realization is a voting system that has a high communication overhead and its management introduces a high computational complexity.

8.2 Review of Some Cryptographic Primitives

In this section, we briefly introduce the cryptographic primitives employed in the design of our authentication scheme.

8.2.1 Secret Sharing Algorithm

The idea of secret sharing is to start with a secret, divide it into pieces called shares, and distribute them amongst a set of users [137]. The pooled shares of specific subsets of users allow the reconstruction of the original secret. We employ a (T, t) threshold secret-sharing algorithm. Such an algorithm generates T shares such that any combination of at least $t \leq T$ shares suffices to reconstruct the original secret. We suggest Shamir's algorithm that generates T distinct shares using the following secret-share generator.

$$\begin{array}{rccl} \mathsf{SSG}_k & : & \mathbb{F} & \longrightarrow & \mathbb{F} \\ & & M & \longmapsto & M + \sum_{i=1}^{t-1} (M \gg i)\, k^i \end{array} \tag{8.1}$$

Here, k is a secret key and $(M \gg i)$ denotes cyclically shifting M to the right i bits. Any combination of t shares generated using distinct secret keys can be used to construct

a system of linearly-independent equations from which the original secret M is uniquely obtained.

8.2.2 Pseudo-random Function

A pseudo-random function is a family of functions with the property that the input-output behavior of a random instance of the family is computationally indistinguishable from that of a random function [16]. The indistinguishability is measured in terms of the ability of a computationally-limited adversary to distinguish the output sequence from a completely randomly generated sequence. A function family is a map $F : \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$, where $\mathcal{K}$ is the set of possible keys, $\mathcal{D}$ is the domain, and $\mathcal{R}$ is the range. For simplicity, we assume $\mathcal{K} = \mathbb{F}$ although this is not a necessary condition. For any $k \in \mathcal{K}$, the instance of the family $F_k : \mathcal{D} \rightarrow \mathcal{R}$ is defined as $F_k(\cdot) := F(k, \cdot)$. Pseudo-random functions can be implemented using the output feedback mode of block ciphers [137]. We employ a family of pseudo-random functions with $\mathcal{K} = \mathbb{F}$, $\mathcal{D} = \mathbb{N} \cup \{ 0 \}$, and $\mathcal{R} = \mathbb{F}$ such that each of them has a unform distribution on the range $\mathcal{R}$.

8.2.3 Hash Tree

Hash trees have many applications in theoretical cryptographic constructions such as *data authentication* and *commitment schemes* [138, 182]. A hash tree on T data values $e_1, \ldots, e_T$ is a binary tree on leaves $H(e_1), \ldots, H(e_T)$, where $H(\cdot)$ is a one-way hash function. Let $\mathcal{U}$ be the set of all nodes of the tree. Every interior node $u \in \mathcal{U}$ has a left child u_L and a right child u_R. By labeling each left child with a "0" and each right child with a "1", the digits along the path from the root identify each node uniquely.

The tree is equipped with a one-way hash function H and a function $\phi : \mathcal{U} \rightarrow \mathbb{F}$ that iteratively assigns a value to every node of the tree. The assignment procedure starts at the leaves of the tree by assigning them arbitrary values of $\mathbb{F}$ that are somehow related to the data values. For every interior node $u \in \mathcal{U}$ with the left and right children u_L and u_R, respectively, the value assigned to u is $\phi(u) := H\big(\phi(u_L) \| \phi(u_R)\big)$.

Assuming that $u_1^l, \ldots, u_T^l$ are the leaves of the tree, we suggest the assignment function with the leaf values $\phi(u_i^l) := H(e_i)$. Every arbitrary leaf is assigned a unique *authentication path* that consists of all the values of all nodes that are siblings of the nodes on the unique path from the root of the tree to that leaf. We note that an authentication path excludes the value of the leaf itself and the root. Therefore, the length of all authentication paths is at most $\lceil \log_2 T \rceil$, where $\lceil \cdot \rceil$ is the ceiling function.

The authentication path of every leaf is used to verify the authenticity of the corresponding data value. Let $\mathsf{AuthPath}(i;\, e_1, \ldots, e_T)$ be an algorithm that calculates the authentication path of the i-th leaf $H(e_i)$. An optimal algorithm is presented in [182] for this purpose that generates the authentication paths in both time and space $O(\log_2 T)$. For every $i \in [\, T\,]$, the data value e_i is authentic if $r = \mathsf{Auth}(e_i, \mathsf{AuthPath}(i;\, e_1, \ldots, e_T))$, where Auth is an algorithm that takes any leaf value along with its corresponding authentication path to generate the root of the tree.

A hash tree for the data values $e_1, \ldots, e_6$ is shown in Figure 8.2. Here, $h_i = H(e_i)$ for all $i \in [\, 6\,]$, $h_{12} = H(h_1 \| h_2)$, $h_{34} = H(h_3 \| h_4)$, $h_{56} = H(h_5 \| h_6)$, $h_{1\sim4} = H(h_{12} \| h_{34})$, and eventually the root value is $r = H(h_{1\sim4} \| h_{56})$. The authentication path for the data value e_3 is the sequence h_4, h_{12}, h_{56}. This data value is authentic if $r = H(H(h_{12} \| H(H(e_3) \| h_4)) \| h_{56})$.

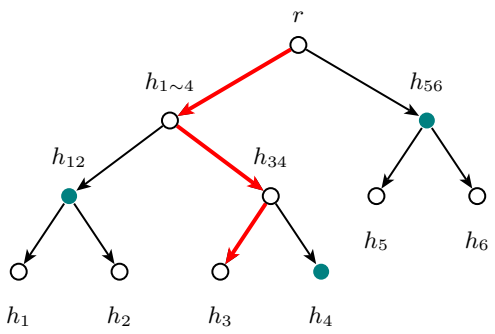


Figure 8.2: Hash tree for four data values.

8.3 Location-Aware Network-Coding Security

Together with my colleague, Mr. Erman Ayday, we proposed LNCS that provides data confidentiality, authenticity, and availability for wireless sensor networks [52, 12]. The proposed scheme makes extensive use of node collaboration to reduce the effect of adversarial activities. To enhance node collaboration and localize the effect of malicious nodes, we divide the terrain into non-overlapping cells with equal shapes. The sensor nodes are densely and uniformly at random deployed in the field. Prior to the network deployment, every node is loaded with a master secret key and a unique ID. We assume that a short period of time after the network deployment, the entire network is secure during which every node obtains the location of its cell and two keys (a cell and a node key) using the location and preloaded information. After the initialization phase, all the nodes delete the secret master key from their memories. Sink is the only entity with the ability of deriving the secret keys of all nodes. We note that this secure initialization period is necessary for location based authentication schemes to generate the location based keys. Hence, the previous location based schemes including the LEDS are also making the secure initialization assumption.

An event in the field is sensed by multiple nodes because of the dense deployment of the sensor nodes. To generate a report, the nodes close to the center of stimulus broadcast their own sensor readings to the neighbors involved in the protocol. (All the inner-cell communications are secured using the cell key.) After the completion of information exchange, all the nodes in the event cell, involved in the protocol, have access to the same set of packets. These nodes aggregate the packets using a secure aggregation function such as median. In the next step, every collaborating node generates a share of the aggregated data using a threshold secret-sharing algorithm and encrypts that using its unique node key. Involved nodes inside the source cell also exchange their encrypted shares between each other. To linearly encode the encrypted shares, these nodes generate the same coefficients matrix using a pseudo-random function. The encoding is performed by multiplying the coefficients matrix to the vector of encrypted secret shares. The involved nodes generate a hash tree on the

encoded packets and the coefficients matrix. Eventually, every node broadcasts only a few encoded packets, the corresponding rows of the coefficients matrix, and the authentication information to the next cell closest to the sink. Every node tags its broadcast with the IDs of its own and its cell. The report is routed in a cell-by-cell fashion on the shortest path toward the sink.

Upon receiving the packets by the nodes in a forwarding cell, these nodes verify the authenticity of the received packets and the coefficients matrix, and drop bogus ones. Similar to the event cell, these nodes generate a common coefficients matrix, encode the authentic packets, and generate a hash tree on the encoded packets and the coefficients matrix. The result is forwarded to the next forwarding cell. Sink is the final check point that verifies the authenticity of the packets. We note that only a fraction of the nodes in every cell take part in the protocol. The remaining nodes remain inactive.

The main contributions of our scheme are summarized in the following.

1. In the proposed scheme, data is forwarded toward the sink using multiple paths and authenticated by multiple nodes, using a collaboration between these nodes. Data authentication is performed without overhearing nodes and voting systems. Such mechanisms, employed by some other schemes, suffer from extensive communication overhead. Moreover, in the proposed scheme, the bogus packets are identified and dropped by the legitimate nodes in the next cell.

2. We employ random network coding in our scheme to generate redundant information that facilitate recovery of the packets erased by the channel or dropped by malicious nodes. This kind of coding significantly improves data availability compared to all other schemes.

3. In contrary to previous schemes, our proposed scheme do not require a trustworthy CH that is responsible for generating the report and forwarding it to the next cell. We emphasize that the existence of a trustworthy CH cannot be guaranteed, and a malicious CH completely breaks down the security of the protocol.

In order to facilitate future references, frequently used notations are listed below with their meanings.

n	Total number of nodes in the network
N	Average number of nodes in every cell
T_0	Number of involved nodes in the event cell
T	Number of involved nodes in the intermediate cells
T'	$(= T_0 + \tau)$ Total number of packets generated after the network coding
$\hat{T}$	Number of legitimate packets after report authentication
t	Minimum number of shares required to reconstruct the message
Δ_i	The ith cell on the forwarding path
$\mathcal{V}_i$	Set of the involved nodes in the cell Δ_i
$\mathbf{e}_i$	Packet vector generated at the cell Δ_i
$\mathbf{C}_i$	Coefficients matrix generated at the cell Δ_i
x	Number of malicious nodes in the entire network
p_{nc}	Fraction of captured nodes in the entire network

8.3.1 General Assumptions

Let n be the total number of sensor nodes in the network and N be the average number of nodes in every cell. An event detected in a cell is endorsed by the collaboration of many nodes within that cell. Next, it is forwarded toward the sink in a cell-by-cell fashion. Our protocol provides a geographical routing mechanism that chooses the shortest path to the sink. Throughout the chapter, we assume $\Delta_0, \Delta_1, \ldots, \Delta_\lambda, \Delta_{\lambda+1}$ is a typical sequence of report forwarding cells starting at the event cell Δ_0 and ending at the sink $\Delta_{\lambda+1}$. In every cell, only a fraction of the nodes are involved in the protocol. For every cell Δ_i, we denote this fraction by the set $\mathcal{V}_i := \left\{ v_1^i, \ldots, v_{T_i}^i \right\}$, where $T_i \leq N$ is the size of the set. In

addition, for simplicity, we assume $T_i =: T$ for all $i \in [\lambda]$, but T_0 is not necessarily equal to T. In other words, the number of involved nodes T_0 in the event cell is not necessarily the same as that in intermediate cells. As we will explain later, this distinction provides robustness in designing the network for required data authenticity and availability. We employ Algorithm 8.1 with $G = N$ and $g = T_i$ to randomly select the set $\mathcal{V}_i$ that consists of nodes with nonzero IDs. A flow chart of the proposed scheme is provided in Figure 8.3.

In the rest of this section, we explain different steps of this scheme.

8.3.2 Setup

This phase takes place prior to the network deployment during which every sensor node is loaded with a unique ID $u \in [n]$ and a secret master key K. In addition, descriptions of the following algorithms are loaded in the memory of every sensor node: a secret-key block cipher Enc_k, a secret-share generator SSG_k as in (8.1), a collusion-resistant hash function H, and a pseudo-random function F_k. Each one of these algorithms is a function $\mathbb{F} \to \mathbb{F}$ and $k \in \mathbb{F}$ is a secret key except the pseudorandom function $F_k : \mathbb{N} \cup \{0\} \to \mathbb{F}$. For the ID assignment, Algorithm 8.1 with $G = g = n$ is employed.

Algorithm 8.1: Tag.

Input: Total number of nodes G and the number of nodes $g \leq G$ to be tagged

Output: An ID in $\{0, 1, \ldots, g\}$ for all G nodes

$\triangleright$ Let $u_1, \ldots, u_G$ be the nodes and $\gamma \geq G$ a fixed integer.

1. For all $i \in [G]$, the node u_i runs a timer initially set to a random value $t_i \in [\gamma]$. Moreover, it sets its counter $c_i \leftarrow 1$.
2. For all $i \in [G]$, the node u_i listens to the medium when its timer fires. If there is no transmission, it considers the value of c_i as its ID and broadcasts it. Otherwise, it sets $c_i \leftarrow c_i + 1$ and defers its transmission.
3. If the value of the last broadcast is $< g$, then return to 2.
4. Other nodes that never get access to the medium, set their IDs to zero.

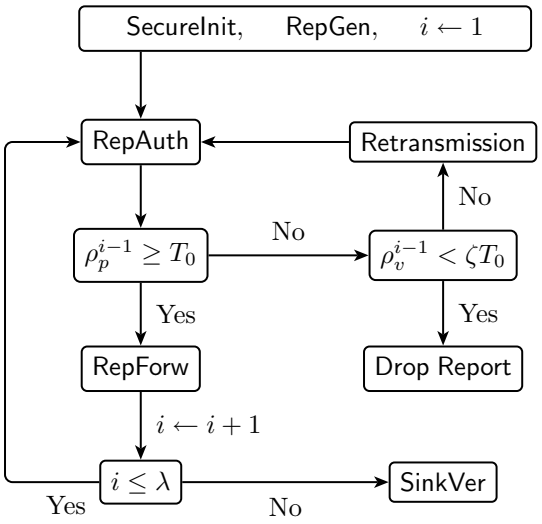


Figure 8.3: Flow chart of the LNCS.

8.3.3 Secure Initialization

The initialization is a short period of time after the network deployment during which we assume there is no adversarial activity. This assumption is practical as it has been made by many other sensor-network protocols.

Assume an arbitrary node u that resides in the cell Δ. Using a localization scheme, such as the one in [123], the node u obtains the location (x_c, y_c) of the center of Δ. The location information is used to derive a *cell key*

$$k_\Delta := H(K \| x_c \| y_c) \tag{8.2}$$

and a *node key*

$$k_u := H(K \| x_c \| y_c \| u) \ . \tag{8.3}$$

These keys are used to secure the inner-cell communications and the report endorsement. At the end of the initialization step, all nodes in the network delete the master key K from their memories.

8.3.4 Report Generation

Triggered by an event or upon a sink query, all the N nodes within the event cell Δ_0, first update their cell key as

$$k_{\Delta_0} \leftarrow H(k_{\Delta_0}) \ . \tag{8.4}$$

(The reason for this update is provided at the end of this subsection.) Then, they run Algorithm 8.1 with $G = N$ and $g = T_0$ to select a subset $\mathcal{V}_0$ consisting of T_0 nodes. The nodes tagged zero by this algorithm do not belong to this subset. Hence, they do not participate in the protocol and remain inactive until the next session. Every node $v_i^0 \in \mathcal{V}_0$ performs the following steps in the specified order.

1. It broadcasts its own sensor reading $M_i \in \mathbb{F}$ to other nodes in the set $\mathcal{V}_0$. (Communications within every cell are secured using the cell key.)

2. Upon the completion of the information exchange, v_i^0 aggregates the T_0 measurements using a resilient aggregation function $\mathcal{A}$. As suggested in [193], *median* is a resilient aggregation function that is a good replacement for the mean value (which is shown to be secure) when the data distribution is symmetric. Let $M \in \mathbb{F}$ be the aggregation value, i.e.,

$$M := \mathcal{A}(M_1, \ldots, M_{T_0}) \ . \tag{8.5}$$

   The aggregated message M may also include a time stamp to avoid message-replay attacks.

3. The node v_i^0 calculates the encrypted share

$$d_i = \mathsf{Enc}_{k_i}(\mathsf{SSG}_{k_i}(M)) \ , \tag{8.6}$$

   where $k_i = k_{v_i^0}$, as in (8.3), is the unique secret key of this node that is derivable only by the sink. Using a (T_0, t) secret sharing scheme allows the sink to reconstruct the message M if up to $T_0 - t$ nodes in $\mathcal{V}_0$ are malicious.

4. The node v_i^0 broadcasts the encrypted share d_i, so all nodes in $\mathcal{V}_0$ have access to the vector $\mathbf{d} := [\, d_1, \ldots, d_{T_0} \,]^\dagger$.

5. Prior to encoding, v_i^0 generates the coefficients matrix $\mathbf{C}_0 = [c_{ij}^0] \in \mathsf{M}_{T',T_0}(\mathbb{F})$ as follows

$$c_{ij}^0 := F_{k_{\Delta_0}}(i \| j) \ , \tag{8.7}$$

where k_{Δ_0} is used as a seed known by all the nodes in $\mathcal{V}_0$ and

$$T' := T_0 + \tau \ , \qquad \tau \geq 0 \ . \tag{8.8}$$

Since F is a pseudo-random function with uniform output distribution, the entries of the matrix $\mathbf{C}_0$ are uniformly at random chosen from $\mathbb{F}$. Hence, the matrix $\mathbf{C}_0$ is invertible with a high probability.

6. The node v_i^0 encodes the vector $\mathbf{d}$ as follows

$$\begin{aligned} \mathbf{e}_0 &:= \mathbf{C}_0 \, \mathbf{d} \in \mathbb{F}^{T'} \\ &= \left[e_1^0, \ldots, e_{T'}^0 \right]^\dagger \ , \end{aligned} \tag{8.9}$$

where $\mathbf{d} := [\, d_1, \ldots, d_{T_0} \,]^\dagger \in \mathbb{F}^{T_0}$. We note that v_i^0 generates more than T packets to compensate for the packets lost or corrupted by noise (due to the medium or adversarial activity) and allow decoding at the sink.

7. The final step of report generation is constructing the hash tree. To evenly distribute the load of handling this step, we split the packet vector $\mathbf{e}_0$ and the rows of the coefficients matrix $\mathbf{C}_0$ into T_0 groups of almost equal sizes. Let $I_1, \ldots, I_{T_0} \subset [\, T' \,]$ be a uniform partition of the set $[\, T' \,]$. For all $i \in [\, T_0 \,]$, the node v_i^0 generates the sequence of authentication paths $\mathbf{a}_{i,1}^0, \ldots, \mathbf{a}_{i,|I_i|}^0$, where

$$\mathbf{a}_{i,j}^0 := \mathsf{AuthPath}(j;\, f_1^0, \ldots, f_{T'}^0) \qquad \forall j \in I_i \ . \tag{8.10}$$

Here, for all $i \in [\, T' \,]$,

$$f_i^0 := e_i^0 \| c_{i1}^0 \| \cdots \| c_{iT_0}^0 \tag{8.11}$$

is the concatenation of the ith packet with only the corresponding rows of the coefficients matrix. We note that both the generated packets and the entries of the coefficients matrix are involved in the hash tree to prevent an adversary from tampering with any one of them.

8. Eventually, the node v_i^0 broadcasts the packets

$$\mathcal{P}_i^0 := \Big(\mathbf{e}_0(I_i), \mathbf{C}_0(I_i), \mathbf{a}_{i,1}^0, \ldots, \mathbf{a}_{i,|I_i|}^0, v_i^0, \Delta_0 \Big) \tag{8.12}$$

to the next forwarding cell. We note that v_i^0 does not transmit the whole packet vector $\mathbf{e}_0$ and the coefficients matrix $\mathbf{C}_0$; it only transmits the rows determined by the index set I_i.

As a summary, the following packets are forwarded from the cell Δ_0 to Δ_1

$$\mathcal{P}_0 := \Big(\mathbf{e}_0, \mathbf{C}_0, \mathbf{a}_{1,1}^0, \ldots, \mathbf{a}_{T,|I_T|}^0, \mathcal{V}_0, \Delta_0 \Big) \quad . \tag{8.13}$$

Upon detecting the reception of the report by the nodes in Δ_1, all the N nodes update their cell key as $k_{\Delta_1} \leftarrow H(k_{\Delta_1})$ and proceed to authenticate the received packets. Updating the cell key adds to the security of the inner-cell communications. In addition, it changes the random selection of the coefficients matrix $\mathbf{C}_0$ prior to every session since the cell key is used as a seed to generate this matrix.

8.3.5 Report Authentication and Filtering

Every nonmalicious node in $\mathcal{V}_0$ transmits approximately $\frac{T'}{T_0}$ packets from the vector $\mathbf{e}_0$. One possible attack is consuming the energy of the nodes in the forwarding cells. To launch such attack, a malicious node in $\mathcal{V}_0$ may transmit many more than $\frac{T'}{T_0}$ packets using the IDs of other nodes in $\mathcal{V}_0$. To prevent this attack, the nodes in $\mathcal{V}_1$ accept at most $\left\lceil \frac{T'}{T_0} \right\rceil$ packets all tagged with the same ID. This threshold for other forwarding cells is $\left\lceil \frac{T'}{T} \right\rceil$.

In order to authenticate packets received from Δ_0, the nodes in $\mathcal{V}_1$ require the root of the hash tree. Since it is not transmitted, they assume it is within the set

$$\mathfrak{R}_0 := \text{mode}\Big\{ \mathsf{Auth}(f_i^0, \mathbf{a}_i^0) \,:\, \forall i \in \big[\, T' \,\big] \Big\} \ , \tag{8.14}$$

where f_i^0 is given in (8.11), $\mathbf{a}_i^0$ is the authentication path of the packet e_i^0, and *mode* is the statistic that from a list of data values returns the ones with the highest repetition. We note that every member of $\mathfrak{R}_0$ is repeated exactly $\rho_p^0 \leq T'$ times which represents the

number of possible authentic packets. For all $i \in [T]$ and $j \in [T']$, the node v_i^1 verifies the authenticity of the packet e_j^0 through the test $\mathsf{Auth}(e_j, \mathbf{a}_j^0) \in \mathfrak{R}_0$. If the packet e_j^0 fails the membership test, it is considered bogus; otherwise, it is authentic. Let $\rho_v^0 \leq T_0$ be the number of nodes in $\mathcal{V}_0$ that have generated all authentic packets. To proceed to the next step, report forwarding, the number of legitimate packets has to be at least T_0 and the number of nonmalicious nodes has to be at least ζT_0, where $0 \leq \zeta \leq 0.5$. (This threshold is ζT for other intermediate forwarding cells.) The possible cases are as follows:

- ♢ $\rho_p^0 \geq T_0$: In this case, any node in the intermediate cell is able to decode the data. Therefore, nodes in $\mathcal{V}_1$ proceed to the report forwarding phase as explained in the following subsection.

- ♢ $\rho_p^0 < T_0$: Based on the value of ρ_v^0, there are two possible cases:

  - $\rho_v^0 \geq \zeta T_0$: This case may happen when malicious nodes, in contradictory to their objective, generate some legitimate packets. Taking advantage of the situation, the nodes in $\mathcal{V}_1$ ask for the retransmission of information from the legitimate nodes in the previous cell Δ_0 and discard all packets transmitted by the nodes detected as malicious.

  - $\rho_v^0 < \zeta T_0$: The report is dropped.

We note that the result of the test $\rho_p^0 \lesseqgtr T_0$ stimulates the necessity for the test $\rho_v^0 \lesseqgtr \zeta T_0$. If $\rho_p^0 \geq T_0$, then the data is decodable in the intermediate cell. Thus, there is no need to check the number of nonmalicious nodes.

Setting $\zeta = 0.5$ implies that the majority of the nodes in the previous forwarding cell have to be nonmalicious to continue report forwarding. In this case, the set $\mathfrak{R}_0$ has at most one element, i.e., there could be only one authentic message. Nevertheless, for $\zeta < 0.5$, the set $\mathfrak{R}_0$ may have more than one element. The implication of this scenario is that there are different reports, each generated by the same number of nodes, but only one of them is authentic. The intermediate nodes cannot determine which report is authentic since making

this decision requires reconstructing the original message from its shares and the keys used to encrypt the shares are unavailable to the intermediate nodes. As we will explain in Section 8.4.3, data availability is inversely related to the value of ζ; for small values of ζ, the probability of data drop due to malicious activities of captured nodes is low. However, as we will see in Section 8.5.2, the payoff for increasing data availability is increasing the communication overhead.

8.3.6 Report Forwarding

Let $J \subseteq [\,T'\,]$ with $|J| = \hat{T} \leq T'$ be the indices of authentic packets after the filtering phase. The nodes in $\mathcal{V}_1$ have access to the common packet-vector $\hat{\mathbf{e}}_0 := \mathbf{e}_0(J) \in \mathbb{F}^{\hat{T}}$ and coefficients matrix $\hat{\mathbf{C}}_0 := \mathbf{C}_0(J) \in \mathsf{M}_{\hat{T},T_0}(\mathbb{F})$. To encode the authentic packets, the nodes in $\mathcal{V}_1$ generate the coefficients matrix $\mathbf{C}'_1 = \left[\, c'^1_{ij} \,\right] \in \mathsf{M}_{T',\hat{T}}(\mathbb{F})$ as follows

$$c'^1_{ij} := F_{k_{\Delta_1}}(i \| j) \ . \tag{8.15}$$

We note that, similar to the event cell, the cell key k_{Δ_1}, known by all the nodes in Δ_1, is used as a seed to randomly generate the matrix $\mathbf{C}'_1$. The next step is performing the network coding and updating the coefficients matrix. For all $i \in [\,T\,]$, the node v_i^1 calculates the packet vector

$$\begin{aligned} \mathbf{e}_1 &:= \mathbf{C}'_1\, \hat{\mathbf{e}}_0 \in \mathbb{F}^{T'} \\ &= \left[\, e_1^1, \ldots, e_{T'}^1 \,\right]^\dagger \end{aligned} \tag{8.16}$$

and updates the coefficients matrix

$$\begin{aligned} \mathbf{C}_1 &:= \mathbf{C}'_1\, \hat{\mathbf{C}}_0 \\ &= \left[\, c_{ij}^1 \,\right] \in \mathsf{M}_{T',T_0}(\mathbb{F}) \ . \end{aligned} \tag{8.17}$$

To evenly distribute the load of generating the authentication information, similar to the event cell, we use a uniform partition $I_1, \ldots, I_T \subset [\,T'\,]$ of the set $[\,T'\,]$. Every node v_i^1 generates the sequence of authentication paths $\mathbf{a}_{i,1}^1, \ldots, \mathbf{a}_{i,|I_i|}^1$, where

$$\mathbf{a}_{i,j}^1 := \mathsf{AuthPath}(j;\, f_1^1, \ldots, f_{T'}^1) \qquad \forall j \in I_i \ . \tag{8.18}$$

Here, $f_i^1 := e_i^1 \| c_{i1}^1 \| \cdots \| c_{iT_0}^1$ for all $i \in [\, T' \,]$. Eventually, the node v_i^1 broadcasts the packets

$$\mathcal{P}_i^1 := \Big(\mathbf{e}_1(I_i),\ \mathbf{C}_1(I_i),\ \mathbf{a}_{i,1}^1,\ \ldots,\ \mathbf{a}_{i,|I_i|}^1,\ v_i^0,\ \Delta_0 \Big) \tag{8.19}$$

to the next forwarding cell. As a summary, the following packets are forwarded from the cell Δ_1 to Δ_2

$$\mathcal{P}_1 := \Big(\mathbf{e}_1,\ \mathbf{C}_1,\ \mathbf{a}_{1,1}^1,\ \ldots,\ \mathbf{a}_{T,|I_T|}^1,\ \mathcal{V}_0,\ \Delta_0 \Big) \ . \tag{8.20}$$

The message forwarding continues in the same fashion at every cell in the sequence $\Delta_1, \ldots, \Delta_\lambda$. It can be easily shown that for every $i \in \{\, 0, 1, \ldots, \lambda \,\}$, we have

$$\mathbf{e}_i = \mathbf{C}_i \, \mathbf{d} \ . \tag{8.21}$$

8.3.7 Sink Verification

The final verification point, the sink, receives the following packets

$$\mathcal{P}_\lambda := \Big(\mathbf{e}_\lambda,\ \mathbf{C}_\lambda,\ \mathbf{a}_{1,1}^\lambda,\ \ldots,\ \mathbf{a}_{T,|I_T|}^\lambda,\ \mathcal{V}_0,\ \Delta_0 \Big) \ . \tag{8.22}$$

Let $\mathfrak{R}_\lambda$, as in (8.14), be the set of possible roots of the hash tree generated at the cell $\Delta_{\lambda-1}$. This implies that the packet vector $\mathbf{e}_\lambda$ consists of $\theta := |\mathfrak{R}_\lambda|$ sub-vectors that are equally likely to be authentic. Let $J_1, \ldots, J_\theta \subset [\, T' \,]$ be the indices of these sub-vectors. From (8.21), we have $\mathbf{e}_\lambda(J_\ell) = \mathbf{C}_\lambda(J_\ell)\, \mathbf{d}_\ell$ for all $\ell \in [\, \theta \,]$, where possibly $\mathbf{d}_\ell = \mathbf{d}$ for only one $\ell \in [\, \theta \,]$. Therefore, for every invertible matrix $\mathbf{C}_\lambda(J_\ell)$, the sink decodes $\mathbf{e}_\lambda(J_\ell)$ as

$$\mathbf{d}_\ell = \big(\mathbf{C}_\lambda(J_\ell) \big)^{-1} \mathbf{e}_\lambda(J_\ell) \ . \tag{8.23}$$

In the next step, the sink decrypts the shares in every $\mathbf{d}_\ell$ using the secret keys of the nodes in $\mathcal{V}_0$. Then, the sink tries to reconstruct the original message using any t out of the T_0 shares. If the reconstructed message is meaningless, the sink tries a different set of t shares. After exhausting all possible combinations, the sink repeats the same process for another vector $\mathbf{d}_\ell$. Therefore, the maximum size of the search space is $\binom{T_0}{t}^\theta$.

8.4 Security Evaluation of the LNCS

In this section, we evaluate the security of our scheme through analytical measurements of the security services provided: confidentiality, authenticity, and availability. Throughout this section, we assume that there are n nodes in the network, and every cell has approximately N nodes. In addition, we assume that an adversary has randomly captured x nodes in the entire network. Therefore, the probability of node capture is $p_{nc} := \frac{x}{n}$.

8.4.1 Data Confidentiality

All the communications within an arbitrary cell Δ are secured using the cell key k_Δ. This key is only used in the event cell to block a passive adversary who is only eavesdropping. Capturing a single node in a cell compromises the security of the entire cell. However, it does not affect other cells since different cells use distinct keys. Even after compromising the security of the event cell, an adversary does not obtain meaningful information. This is because the shares generated at the event cell are encoded using the unique keys pairwise between the report-generating nodes and the sink.

The data confidentiality of the LNCS is the same as that in LEDS proposed in [166]. A cell is compromised when at least one node inside that cell is captured. Therefore, the probability P_{comp} of cell compromise with respect to data confidentiality is

$$P_{comp} = 1 - \frac{\binom{n-N}{x}}{\binom{n}{x}} \quad . \tag{8.24}$$

The curves of this probability are provided in [166].

8.4.2 Data Authenticity

One possible attack launched by an adversary is capturing enough number of nodes in the event cell to forge a report. We note that the shares of an event are generated at the event cell using the secret keys known only to the report endorsing nodes and the sink. Therefore, an adversary is unable to deceive the sink by capturing nodes along the forwarding path.

Since the sink requires at least t consistent packets to reconstruct the data, the adversary has to capture at least t nodes within the event cell. Thus, the probability of data authenticity is

$$p_{auth} = \sum_{j=0}^{t-1} p_c(j) \ , \tag{8.25}$$

where $p_c(j)$ is the probability that exactly j random nodes in the event cell are captured, i.e.,

$$p_c(j) = \frac{\binom{N}{j}\binom{n-N}{x-j}}{\binom{n}{x}} \ , \qquad j = 0, 1, \ldots, N \ . \tag{8.26}$$

The probability of authenticity is plotted in Figure 8.4 for different values of N and t. In this graph, p_{nc} is the probability of node capture. As these curves show, increasing the value of t improves P_{auth} since the number of nodes to be captured by an adversary also increases. Another observation is that increasing the cell size degrades the probability of authentication. This is because in a large cell, the probability that a randomly captured node resides in the cell under study is high. As an example, for $t = 40$, the probability of authenticity is 75% when 36% of the nodes are captured.

8.4.3 Data Availability

To prevent the sink from receiving a legitimate report, an adversary has to capture a minimum number of involved nodes in an arbitrary forwarding cell Δ_i. As explained in

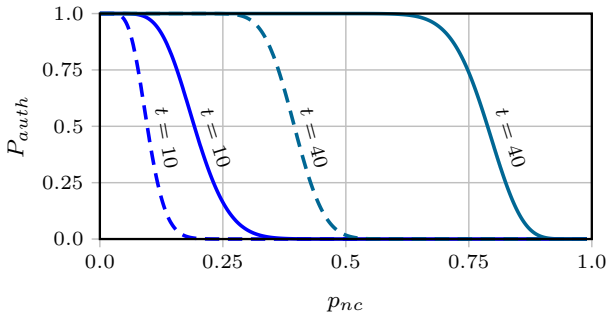


Figure 8.4: Probability of authenticity in a network of size n = 10,000 and cell sizes N = 50 (solid lines) and N = 100 (dashed lines).

Section 8.3.5, ζT_i is the threshold[2] on the number of nonmalicious nodes detected in $\mathcal{V}_{i-1}$ that are required by the cell Δ_i to forward the message to the cell Δ_{i+1}. Therefore, the adversary has to capture at least $T-\zeta T_i+1$ involved nodes from the set $\mathcal{V}_i$. In light of this observation, the probability of data availability is

$$P_{av}^{i} = \sum_{j=0}^{\lceil T-\zeta T_i \rceil+1} p_{inv}^{i}(j) \ , \tag{8.27}$$

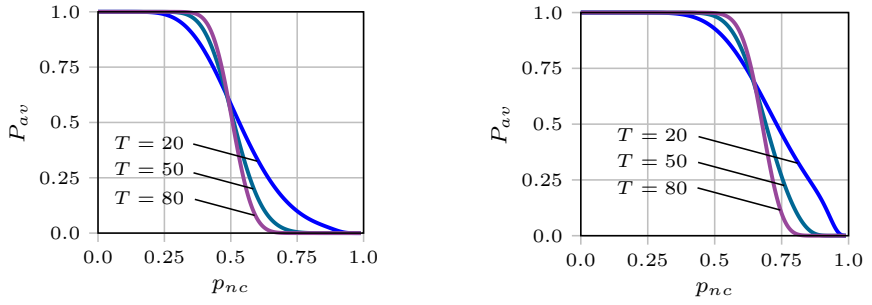
where $p_{inv}^{i}(j)$ is the probability that among the nodes captured in the cell Δ_i, exactly j of them are involved. Using conditional probability, one can easily show that

$$p_{inv}^{i}(j) = \sum_{\ell=j}^{N-T_i+j} p_c(\ell) \binom{\ell}{j} \left(\frac{T_i}{N}\right)^{j} \left(1-\frac{T_i}{N}\right)^{\ell-j} \ . \tag{8.28}$$

A possible attack is selective forwarding in which malicious nodes may refuse to forward the report and simply drop it [112]. In our proposed scheme, this attack fails when an adversary randomly captures a few nodes within a forwarding cell. The adversary achieves her goal by capturing only involved nodes in a cell.

Assuming $T_0 = T$, the probability of data availability P_{av} is plotted in Figure 8.5 for different values of T and ζ. In all these curves, a general observation is that for small values of p_{nc}, increasing T improves the probability P_{av} because the adversary has to capture more nodes. However, beyond an specific value of p_{nc} this effect reverses, i.e., increasing T decreases the probability P_{av}. This phenomenon becomes clear recalling that the data

[2]We recall that $T_i = T$ for all $i \geq 1$.



(a) $\zeta = \frac{1}{2}$. (b) $\zeta = \frac{1}{3}$.

Figure 8.5: Probability of availability in a network with $n = 10{,}000$ and $N = 100$.

is available in a forwarding cell only when this cell has received authentic packets from at least ζT_0 nonmalicious nodes in the previous cell. When there are too many malicious nodes in the network, finding at least ζT nonmalicious nodes becomes difficult for large values of T. Another observation is that for a fixed T, increasing N degrades availability since the probability that a node is involved decreases. Another observation is that decreasing ζ improves the availability since ζT is the threshold on data availability.

To mention a few numerical examples, in Figure 8.5a, at the crossing point of all curves, data availability is 60% when 50% of the nodes are captured. For the same number of malicious nodes and $T = 20$, in Figure 8.5b, data availability improves to 93%.

8.5 Performance Evaluation of the LNCS

In this section, we evaluate the performance of our scheme in terms of computation and communication overheads per sensor node. Moreover, as explained in Section 8.3.5, a forwarding cell may request the retransmission of the report from the previous cell. Hence, we calculate the probability of retransmissions that is considered communication overhead. Throughout this section, we assume $T' = O(T_0)$ that is a feasible assumption in network coding.

8.5.1 Computation Overhead

The first phase in our scheme is report generation. The generation of the matrix $\mathbf{C}_0$ as in (8.7) and the calculation of the vector $\mathbf{e}_0$ in (8.9) are computationally the most expensive calculations in this phase. They both cost $O(T_0^2)$ that is the total computational complexity of report generation. We note that data aggregation in (8.5) is usually a fast operation. For example, the computational complexity of calculating median, as suggested in Section 8.3.4, is $O(T_0 \log_2 T_0)$ [40].

The next phase is report authentication and filtering. The only computation performed in this phase is constructing the set $\mathfrak{R}_i$ that consists of the mode of T' data values. This is

a relatively cheap operation with complexity $O(\log_2 T_0)$.

The last phase performed by the sensor nodes is report forwarding. The most expensive computation in this phase is calculating the matrix $\mathbf{C}_i$ as in (8.17) that costs $T' \hat{T} T_0 = O(T_0^3)$. Finally, we conclude that the computational complexity of our scheme is $O(T_0^3)$ per sensor node.

8.5.2 Communication Overhead

In this subsection, we calculate the communication overhead per sensor node in terms of the number of elements of $\mathbb{F}$ transmitted or received considering the fact that both data transmission and reception consume the same amount of energy.

During the report generation phase, every node in the set $\mathcal{V}_0$ broadcasts its own sensor reading to other nodes in that set. Since the nodes outside this set remain inactive, the communication overhead of this operation is exactly T_0 per node. At the end of report generation, every node v_i^0 transmits the set of packets $\mathcal{P}_i^0$ as in (8.12) to the next cell. The number of packets in this set approximately is $\frac{T'}{T_0}(1 + \log_2 T') + T' = O(T_0)$. Therefore, the communication overhead of report generation is $O(T_0)$ per node.

Every node in a forwarding cell receives a set of packets as in (8.13) that approximately consists of $T' + T' T_0 + T \frac{T'}{T_0} \log_2 T' = O(T_0^2)$ packets. In addition, every such node transmits a set of packets as in (8.19) that, similar to the report generation phase, consists of $O(T_0)$ packets. Therefore, we conclude that in our scheme, the communication overhead per node is $O(T_0^2)$.

8.5.3 Retransmission

As explained in Section 8.3.5, the nodes in a forwarding cell Δ_{i+1} may require the retransmission of information from the previous cell. The retransmission occurs only when the number of nonmalicious nodes detected in the previous cell ρ_v^i is strictly less than ζT_i while the number of authentic packets ρ_p^i is greater than or equal to T_0. We recall that a nonmalicious node in $\mathcal{V}_i$ generates approximately $\frac{T'}{T_i}$ authentic packets. Therefore, to violate the

threshold T_0 on the number of authentic packets ρ_p^i, the adversary has to capture at least

$$\eta_i := \left\lfloor T_i \left(1 - \frac{T_0}{T'}\right) \right\rfloor + 1 \tag{8.29}$$

nodes from $\mathcal{V}_i$. On the other hand, to request retransmission, there has to be at least ζT_i nonmalicious nodes in Δ_i, which implies that the adversary has to capture not more than $T_i(1-\zeta)$ nodes in Δ_i. Considering these facts, retransmission may happen only when $\eta_i < T_i(1-\zeta)$, i.e.,

$$T_0 > \zeta(T_0 + \tau) \tag{8.30}$$

by (8.8). In this case, the probability of retransmission requested by the nodes in Δ_{i+1} is

$$P_{re}^{i+1} := \sum_{j=\eta_i}^{\lfloor T_i(1-\zeta) \rfloor} p_{inv}^i(j) \ . \tag{8.31}$$

Here, $p_{inv}^i(j)$, given in (8.28), is the probability that exactly j involved nodes in the cell Δ_i are captured.

The probability of retransmission for different ratios of over-transmission $\frac{\tau}{T}$ is plotted in Figure 8.6. As the curves in this figure show, increasing over-transmission decreases the probability of retransmission, which intuitively makes sense. It can also be mathematically explained noting that by increasing τ, the threshold η in (8.29) increases as well. Another observation is that when the fraction of captured nodes in the network is high, the probability of retransmission is low. Although practically of less interest, this situation happens

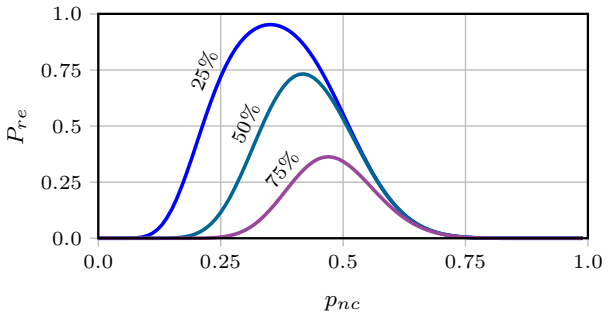


Figure 8.6: Probability of retransmission in a network of size $n = 10{,}000$ and other parameters $N = 100$, $T_0 = T = 50$, $\zeta = 0.5$, and $\frac{\tau}{T} \in \{\, 0.25,\, 0.5,\, 0.75 \,\}$.

when the large number of captured nodes causes the report drop and the breakdown of the protocol.

8.6 Comparison with LEDS

In this section, we compare LNCS with LEDS in terms of security and overhead since LEDS is the only scheme that provides data availability. We note that none of the other schemes (IHA, SEF, and LBRS) provides data availability since data is transmitted on a path, consisting of single nodes, toward the sink. Therefore, a malicious node on the path may drop the report to prevent its reception by the sink. In the following, we provide a comparison between LNCS and LEDS.

1. The transmission of data from one cell to another is performed by a single trustworthy node in LEDS called CH. The existence of such node cannot be guaranteed. In LNCS, every node involved in the protocol broadcasts part of the generated report. Thus, in terms of reliability in data transmission, LNCS outperforms LEDS.

2. To provide collaboration between overhearing nodes in LEDS, excessive amount of redundant communication between adjacent cells is necessary to collect the votes of the nodes in the previous cell on the broadcast message. Moreover, this voting mechanism is not practical and will fail even in the presence of a few malicious nodes. The LNCS does not employ a voting system. Therefore, it does not bear with the communication overhead required for such a system.

3. In LEDS, the nodes in a forwarding cell behave independently. Therefore, malicious nodes cause serious data availability and authenticity problems. For example, a malicious node in LEDS may take the role of the CH and modify the legitimate message. The use of network coding in our scheme significantly improves data availability.

   In Figure 8.7, we compare LNCS with LEDS in terms of data availability. In this experiment, the number of involved nodes in every cells is 40. Since in LEDS, all the

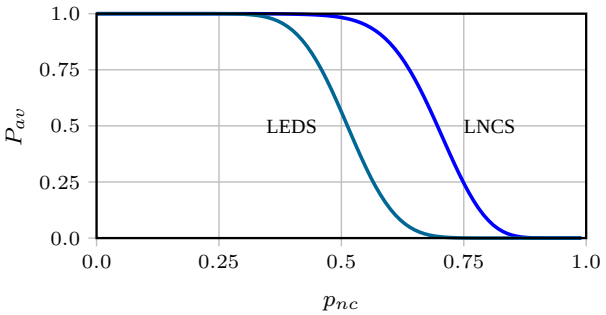


Figure 8.7: Comparing data availability between LNCS and LEDS. The network size is $n =$ 10,000 and other parameters are $N = 50$, $T_0 = T = 40$, and $\zeta = \frac{1}{3}$. In LEDS, we have assume $t = 20$ and the number of nodes per cell is 40.

nodes in every forwarding cell participate in the protocol, we have assumed there are 40 nodes in every cell for a fair comparison. The other assumption we have made is $t = \frac{T}{2}$ that provides a fair tradeoff between data availability and authenticity. As the figure shows, data availability in LNCS is much higher than that in LEDS. For example, when 50% of the nodes in the entire network are compromised, the probabilities of data availability in LNCS and LEDS are 98% and 56%, respectively. The payoff for increasing data availability in LNCS is the increase in communication overhead.

4. The coefficients matrix used for network coding in LNCS is transmitted from one cell to another. Therefore, in terms of communication overhead, the LEDS outperforms LNCS. We note that communication overhead is the intrinsic drawback of all networks using random network coding.

8.7 Summary

In this chapter, we proposed a package of security services for wireless sensor networks as a protocol named location-aware network-coding security (LNCS). As the name of the protocol implies, the nodes take advantage of the location information by dividing the terrain into non-overlapping cells and deriving location binding keys during the secure initialization phase. Since a malicious cluster head can completely break down the security of a proto-

col, in LNCS, we have remedied the need to a cluster head that is responsible for report generation and forwarding. An event detected in the field is sensed by several nodes and aggregated by all of them. Using a secret sharing algorithm, the aggregated information is divided into several shares that are forwarded toward the sink in a cell-by-cell fashion. To provide data availability, we employed random network coding in our scheme. A comparison with other schemes showed a significant improvement in data availability. As an authentication mechanism, we construct a hash tree on the encoded packets generated at every cell. The packets that fail the authentication test are dropped. Every node in the forwarding cell transmits only a fraction of the generated packets along the corresponding authentication information. The sink is the final entity being able to reconstruct the original message using a few shares of the message. A comparison with the previous schemes revealed significant improvement in data availability while maintaining the same level of data confidentiality and authenticity.

Part IV

Postliminary Material

CHAPTER 9

Conclusion of the Thesis

♢ Chapter 2 briefly reviewed univariate orthogonal wavelets over fields of characteristic two, their filter-bank realization, and the unitary and paraunitary (PU) building blocks. Considering the importance of PU matrices in the design of orthogonal filter banks, Chapter 3 undertook the study of the factorization of bivariate PU matrices over fields of characteristic two. In this chapter, we proposed a two-level factorization method. In this method, a bivariate matrix polynomial is considered a univariate one in one of the variables with its coefficients being matrix polynomials themselves in the other variable. By relaxing the definition of PU matrices, we successfully factorized bivariate matrix polynomials one level over the ring of polynomials into the product of fully-parameterized ring building blocks that we have proposed. A second level of factorization is sometimes possible depending on the factors obtained in the first level.

♢ Multivariate cryptography using wavelet transform and PU matrices is the topic of Part II. We proposed an iterative self-synchronizing stream cipher, named WSSC, using the wavelet transform as a sequence transformer and also some nonlinear components in Chapter 5. Although the structure of the WSSC mimics the canonical representation of self-synchronizing stream ciphers, there some fundamental differences. For example, the secret key determines the wavelet coefficients in contrast to the traditional design in which it determines initial states of the shift registers. The cryptanalysis of the new cipher, where we studied many algebraic attacks, revealed that two rounds of the WSSC resists all attacks. In addition, we studied the circuit complexity of our cipher and compared it with that in the AES. This comparison shows that the WSSC has less

circuit complexity to the cost of more memory.

In Chapter 6, we proposed a new framework for the design of public-key cryptosystems. This new approach is based on the fact that the columns of any PU matrix are bases for the module of polynomial vectors. We took advantage of this fact to design a trapdoor one-way function based upon we proposed new public-key and digital signature schemes. Using a mathematical conjecture, we provided evidence connecting the security of these new designs to the difficulty of solving systems of multivariate polynomial equations, which is considered to be hard problem. Public information in both cryptosystems are several multivariate polynomials. The encryption and signature verification in the new schemes are very efficient since they both require only evaluations of the public polynomials. Because of their low computational complexity, these new cryptosystems are suitable for applications in resource-limited devices. In the cryptanalysis of these systems, we considered many algebraic attacks. Results showed no vulnerability to any one of these attacks.

◊ Providing security services such as data confidentiality, integrity, and availability to wireless sensor networks (WSNs) was the problem under study in Part III. Data confidentiality is provide through the use of symmetric cryptography that requires a secret key shared between the communicating parties. Because of the low computational power of sensor nodes, we proposed a key pre-distribution method named MKPS in Chapter 7. This method uniquely assigns IDs to all sensors using points on a virtual hypercube in the d-dimensional space for some fixed integer d. Based on these IDs, exactly $d-1$ symmetric multivariate polynomials are selected from a pool of such polynomials and their univariate shares are stored in the memory of sensor nodes prior to their deployment in the field. After the deployment, every two adjacent sensor nodes that their IDs are at the unit Hamming distance of each other use the polynomial shares stored in their memories to establish exactly $d-1$ common keys. The final key is a symmetric combination of the common keys. We showed that this method significantly improves

the resiliency of the network to the physical capture of nodes by an adversary. This improvement comes to the cost of reducing the probability of connectivity, which can be compensated by increasing the communication radius. We also provided an algorithm to optimize the dimension d based on two metrics: the network connectivity and the network resiliency.

Taking advantage of the node location information (when it is available), we proposed location-aware MKPS or LA-MKPS. In this scheme, the entire terrain is divided into non-overlapping hexagonal cells. Sensor nodes are uniformly at random scattered in every cell. The inner- and intra-cell communications are secured using an MKPS and a bivariate version of that, respectively. The LA-MKPS provides perfect connectivity while inheriting the superior resiliency of the MKPS.

◊ In Chapter 8, we proposed a new scheme called LNCS that provides data authenticity and availability to WSNs. This scheme divides the entire terrain into non-overlapping hexagonal cells inside each of which nodes are uniformly at random scattered. Using the location of the center of their residing cell, all sensor nodes derive two keys, a cell key and a node key, from a master key pre-stored in them. Triggered by an event in the field or upon a query by the sink, a report is collaboratively compiled by the nodes in a cell. Using a secret sharing scheme and the node keys, collaborating nodes generate secret shares of the aggregated sensed data. To provide data availability, we employed random network coding. The secret shares are encoded by multiplying their corresponding vector by a matrix of random coefficients. For authenticity, we used a hash tree constructed on the encoded data. The final report is forwarded toward the sink on a cell-by-cell fashion. At every cell, the authenticity of the received data is verified via the hash tree and unverified packets are considered bogus and dropped enroute. Sink is the final point where the data is verified, decoded, and reconstructed. In a comparison with previous schemes, we showed that the LNCS outperforms in terms of the data availability.

APPENDICES

APPENDIX A

Proofs of Chapter 3 in Part I

1 Proof of Lemma 3.1.1

By the PU property of $\mathbf{P}(x, y)$, i.e.,

$$\left[\sum_{i=0}^{L} \mathbf{p}_{i,0}^{\dagger}\, x^i + \sum_{i=0}^{L} \mathbf{p}_{i,1}^{\dagger}\, x^i\, y\right] \left[\sum_{i=0}^{L} \mathbf{p}_{i,0}\, x^{-i} + \sum_{i=0}^{L} \mathbf{p}_{i,1}\, x^{-i}\, y^{-1}\right] = \mathbf{I} \ ,$$

we conclude that the coefficients of the terms $x^L y$, $x^L y^{-1}$, and x^L must be zero:

$$\mathbf{p}_{L,1}^{\dagger}\, \mathbf{p}_{0,0} = 0 \tag{A.1a}$$

$$\mathbf{p}_{L,0}^{\dagger}\, \mathbf{p}_{0,1} = 0 \tag{A.1b}$$

$$\mathbf{p}_{L,0}^{\dagger}\, \mathbf{p}_{0,0} + \mathbf{p}_{L,1}^{\dagger}\, \mathbf{p}_{0,1} = 0 \ . \tag{A.1c}$$

Since $\mathbf{P}^{\dagger}(x, y)$ is also PU, using a similar approach and setting the coefficients of the same terms, one gets:

$$\mathbf{p}_{0,0}\, \mathbf{p}_{L,1}^{\dagger} = \mathbf{0} \tag{A.2a}$$

$$\mathbf{p}_{0,1}\, \mathbf{p}_{L,0}^{\dagger} = \mathbf{0} \tag{A.2b}$$

$$\mathbf{p}_{0,0}\, \mathbf{p}_{L,0}^{\dagger} + \mathbf{p}_{0,1}\, \mathbf{p}_{L,1}^{\dagger} = \mathbf{0} \ . \tag{A.2c}$$

By the statement of the lemma, $\mathbf{p}_{L,0}$ and $\mathbf{p}_{L,1}$ are not both zero. Hence, we consider the following cases.

- $\underline{\mathbf{p}_{L,0} \neq \mathbf{0},\ \mathbf{p}_{L,1} = \mathbf{0}}$: In this case, equations (A.1) simplify to $\mathbf{p}_{L,0}^{\dagger}\, \mathbf{p}_{0,0} = 0$ and $\mathbf{p}_{L,0}^{\dagger}\, \mathbf{p}_{0,1} = 0$. Equations (3.5a) are satisfied by taking the vector $\mathbf{v}$ as any one of the nonzero columns of $\mathbf{p}_{L,0}$.

- $\underline{\mathbf{p}_{L,1} \neq \mathbf{0},\ \mathbf{p}_{L,0} = \mathbf{0}}$: Similar to the previous case, equations (A.1) simplify to $\mathbf{p}_{L,1}^{\dagger}\,\mathbf{p}_{0,0} = 0$ and $\mathbf{p}_{L,1}^{\dagger}\,\mathbf{p}_{0,1} = 0$. Equations (3.5a) are satisfied by taking the vector $\mathbf{v}$ as any one of the nonzero columns of $\mathbf{p}_{L,1}$.

- $\underline{\mathbf{p}_{L,0}^{\dagger}\,\mathbf{p}_{0,0} = \mathbf{0}}$ It can be easily verified (using (A.1)) that this case is the same as the previous case.

- $\underline{\mathbf{p}_{L,0}^{\dagger}\,\mathbf{p}_{0,0} \neq \mathbf{0},\ \mathbf{p}_{L,1} \neq \mathbf{0}}$: We, first, post-multiply (A.2c) by $\mathbf{p}_{00}$ to get

$$\mathbf{p}_{0,0}\,\mathbf{p}_{L,0}^{\dagger}\,\mathbf{p}_{0,0} + \mathbf{p}_{0,1}\,\mathbf{p}_{L,1}^{\dagger}\,\mathbf{p}_{0,0} = \mathbf{0}\ .$$

  By (A.1a), this equation simplifies to $\mathbf{p}_{0,0}\,\mathbf{p}_{L,0}^{\dagger}\,\mathbf{p}_{0,0} = \mathbf{0}$. Post-multiplying (A.2b) by $\mathbf{p}_{0,0}$, we get $\mathbf{p}_{0,1}\,\mathbf{p}_{L,0}^{\dagger}\,\mathbf{p}_{0,0} = \mathbf{0}$. With these two results, equations (3.5b) are satisfied by taking $\mathbf{v}$ as any one of the nonzero columns of $\mathbf{p}_{L,0}^{\dagger}\,\mathbf{p}_{0,0}$. ∎

2 Proof of Lemma 3.2.1

Let the polynomial vector

$$\mathbf{v}(x) = \begin{bmatrix} f(x) \\ g(x) \end{bmatrix}$$

be self-orthogonal, i.e.,

$$f(x)f(x^{-1}) = g(x)g(x^{-1})\ . \tag{A.3}$$

There always exists an integer $\ell \in \mathbb{Z}$ such that

$$f(x) = x^{\ell}\underbrace{(a_0 + \cdots + a_n x^n)}_{f_+(x)}\ , \tag{A.4}$$

where $n \in \mathbb{Z}_{\geq 0}$ and $a_i \in \mathbb{F}$ for all $i = 0, 1, \ldots, n$. The polynomial $f_+(x)$ in (A.4) can be factored into the product of irreducible polynomials $f_i(x) \in \mathbb{F}[x]$, i.e.,

$$f_+(x) = \prod_{i=1}^{N} f_i(x) \tag{A.5}$$

for some $N \in \mathbb{N}$. Using (A.5) and (A.4) in (A.3), we get

$$g(x)g(x^{-1}) = \prod_{i=1}^{N} f_i(x) \prod_{j=1}^{N} f_j(x^{-1}) \ .$$

Since the polynomials $f_1(x), \ldots, f_N(x)$ are all irreducible, for every $i \in [\, N \,]$, the factor $f_i(x)$ belongs to either $g(x)$ or $g(x^{-1})$. In either case, the factor $f_i(x^{-1})$ belongs to the other one. Thus, $g(x)$ can be expressed as

$$g(x) = h(x) \prod_{i \in \mathcal{I}^+} f_i(x) \prod_{j \in \mathcal{I}^-} f_j(x^{-1}) \ , \tag{A.6}$$

where $\{\, \mathcal{I}^+, \mathcal{I}^- \,\}$ is a partition of $[\, N \,]$, i.e., $\mathcal{I}^+ \cap \mathcal{I}^- = \varnothing$ and $\mathcal{I}^+ \cup \mathcal{I}^- = [\, N \,]$. In (A.6), $h(x)$ has to satisfy the constraint $h(x)\, h(x^{-1}) = 1$. It is easy to show that this constraint enforces $h(x)$ to be a monomial. Let $h(x) = x^s$ for some $s \in \mathbb{Z}$. Using these results, the polynomial $f(x)$ can be reformulated as

$$f(x) = x^{\ell} \prod_{i \in \mathcal{I}^+} f_i(x) \prod_{j \in \mathcal{I}^-} f_j(x) \ . \tag{A.7}$$

Let $\beta(x) = x^s \prod_{i \in \mathcal{I}^+} f_i(x)$. By (A.6) and (A.7), we obtain

$$\mathbf{v}(x) = \beta(x) \begin{bmatrix} x^m p(x) \\ p(x^{-1}) \end{bmatrix} \ ,$$

where $p(x) = \prod_{i \in \mathcal{I}^-} f_i(x)$ and $m = \ell - s$. ∎

3 Proof of Lemma 3.2.2

Suppose the polynomial vector

$$\begin{bmatrix} f(x) \\ g(x) \end{bmatrix} \ \in \ (\mathbb{F}[x^{\pm 1}])^2$$

is orthogonal to $\mathbf{v}(x)$. By the orthogonality condition, we must have

$$x^{-m}\, p(x^{-1})\, f(x) = p(x)\, g(x) \ . \tag{A.8}$$

Since $x^{-m}p(x^{-1})$ and $p(x)$ are relatively prime, we obtain

$$f(x) = x^m\, p(x)\, \beta(x) \ , \qquad g(x) = p(x^{-1})\, \beta(x) \ , \tag{A.9}$$

which is the desired result. ∎

4 Proof of Lemma3.4.1

By (3.13), all column vectors of $\mathbf{P}_L(x)$ are in $\mathsf{Null}_R(\mathbf{P}_0(x))$, which is not self dual by the assumption of the lemma. Hence, at least one of the column vectors of $\mathbf{P}_L(x)$ is non self-orthogonal. Let $\mathbf{v}(x)$ be such vector. Obtain $\mathbf{P'}(x,y)$ as

$$\begin{aligned}
\mathbf{P'}(x,y) &:= \mathsf{B}_1^{\dagger}\Big(y;\, \mathbf{v}(x)\Big)\, \mathbf{P}(x,y) \\
&= \mathbf{v}(x)\mathbf{v}^{\dagger}(x)\mathbf{P}_0(x)y \\
&\quad + \sum_{i=0}^{L-1} \Big[\alpha(x)\mathbf{P}_i(x) + \mathbf{v}(x)\mathbf{v}^{\dagger}(x)\mathbf{P}_i(x) + \mathbf{v}(x)\mathbf{v}^{\dagger}(x)\mathbf{P}_{i+1}(x)\Big]\, y^{-i} \\
&\quad + \Big[\alpha(x)\mathbf{P}_L(x) + \mathbf{v}(x)\mathbf{v}^{\dagger}(x)\mathbf{P}_L(x)\Big]\, y^{-L} \ .
\end{aligned} \tag{A.10}$$

The only term on the right hand side with positive y-exponent is $\mathbf{v}(x)\mathbf{v}^{\dagger}(x)\mathbf{P}_0(x)y$. Nevertheless, this term vanishes since $\mathbf{v}(x) \in \mathsf{Null}_R(\mathbf{P}_0(x))$. Since $\mathbf{P'}(x,y)$ is obtained through the multiplication of two PU matrices, it is PU over $\mathbb{F}[x^{\pm 1}]$ as well.

$$\mathbf{P'}^{\dagger}(x,y)\, \mathbf{P}(x,y) = \alpha^2(x)\, \mathbf{I} \tag{A.11}$$

Pre-multiplying both sides of (A.10), we get the desired result. After taking the determinant of both sides of (A.10) and using Lemma 3.3.1, we obtain

$$\mathsf{det}\, \mathbf{P'}(x,y) = [\mathsf{det}\, \mathbf{P}(x,y)]\ \alpha^{-2}(x)\, y \ , \tag{A.12}$$

which implies the degree of $\mathbf{P}(x,y)$ with respect to y is reduced by one. ∎

5 Proof of Theorem 3.4.1

Since both null spaces of $\mathbf{P}_0(x)$ are self dual, by Fact 3.2.2, it is in the form

$$\mathbf{P}_0(x) = \beta(x) \underbrace{\begin{bmatrix} x^m\, p(x) \\ p(x^{-1}) \end{bmatrix}}_{\mathbf{u}(x)} \underbrace{\begin{bmatrix} x^{-r}\, q(x^{-1}) & q(x) \end{bmatrix}}_{\mathbf{v}^\dagger(x)} , \tag{A.13}$$

where $\beta(x), p(x), q(x) \in \mathbb{F}[x^{\pm 1}]$ and $m, r \in \mathbb{Z}$. Suppose $\mathbf{P}(x, y)$ cannot be factored as in (3.15). Then, for any $\mu(x) \in \mathbb{F}[x^{\pm 1}]$, the matrix polynomial $\mu(x)\,\mathbf{P}(x, y)$ cannot be factorable either. By (3.12), we can expand $\mu(x)\,\mathbf{P}(x, y)$ as

$$\mu(x)\,\mathbf{P}(x, y) = \gamma_0(x)\,\mathbf{u}(x)\,\mathbf{v}^\dagger(x) + \sum_{i=1}^{L} \mathbf{P}'_i(x)\, y^{-i} , \tag{A.14}$$

where

$$\gamma_0(x) := \mu(x)\beta(x) \tag{A.15}$$

and $\mathbf{P}'_i(x) := \mu(x)\,\mathbf{P}_i(x)$ for all $i \in [\,L\,]$. As the first step, we attempt to extract the building block $\mathsf{S}_2\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big)$. Let

$$\mathbf{A}(x) := \begin{bmatrix} x^{m-r} p(x) q(x^{-1}) & 0 \\ 0 & p(x^{-1}) q(x) \end{bmatrix} .$$

In addition, let $\mathbf{F}_1(x, y)$ and $\mathbf{F}_2(x, y)$ be the polynomial matrices obtained by pre- and post-multiplying $\mu(x)\mathbf{P}(x, y)$ by $\mathsf{S}_2^\dagger\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big)$, respectively.

$$\begin{aligned} \mathbf{F}_1(x, y) &:= \mathsf{S}_2^\dagger\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big)\; \mu(x)\mathbf{P}(x, y) \\ &= \underbrace{\Big[\gamma_0(x)\mathbf{A}^\dagger(x)\mathbf{u}(x)\mathbf{v}^\dagger(x) + \zeta(x)\mathbf{v}(x)\mathbf{u}^\dagger(x)\mathbf{P}'_1(x)\Big]}_{\mathsf{G}_1(x)}\; y + \sum_{i=0}^{L} \mathbf{F}_{1,i}(x)\, y^{-i} \end{aligned} \tag{A.16a}$$

$$\begin{aligned} \mathbf{F}_2(x, y) &:= \mu(x)\mathbf{P}(x, y)\; \mathsf{S}_2^\dagger\Big(y;\, \mathbf{u}(x), \mathbf{v}(x), \zeta(x)\Big) \\ &= \underbrace{\Big[\gamma_0(x)\mathbf{u}(x)\mathbf{v}^\dagger(x)\mathbf{A}^\dagger(x) + \zeta(x)\mathbf{P}'_1(x)\mathbf{v}(x)\mathbf{u}^\dagger(x)\Big]}_{\mathsf{G}_2(x)}\; y + \sum_{i=0}^{L} \mathbf{F}_{2,i}(x)\, y^{-i} \end{aligned} \tag{A.16b}$$

Both $\mathbf{F}_1(x, y)$ and $\mathbf{F}_2(x, y)$ are PU over $\mathbb{F}[x^{\pm 1}]$ since they are obtained by multiplying polynomial matrices with the same property. If there exists a symmetric polynomial $\zeta(x)$ such

that either $\mathbf{G}_1(x) = \mathbf{0}$ or $\mathbf{G}_2(x) = \mathbf{0}$, we have successfully extracted a degree-two building block. Since by our assumption, $\mu(x)\mathbf{P}(x, y)$ is not factorable, none of these matrices must be zero. Moreover, using the equality $\mathbf{A}^\dagger(x)\mathbf{u}(x)\mathbf{v}^\dagger(x) = p(x)p(x^{-1})\mathbf{v}(x)\mathbf{v}^\dagger(x)$ (which can be easily verified), we can show that

$$\mathbf{G}_1(x) = p(x)p(x^{-1})\mathbf{v}(x)\Big[\gamma_0(x)\mathbf{v}^\dagger(x) + \zeta'(x)\underbrace{\mathbf{u}^\dagger(x)\mathbf{P}'_1(x)}_{\mathbf{e}_1^\dagger(x)}\Big] \ , \tag{A.17}$$

where $\zeta'(x) \in \mathbb{F}[x^{\pm 1}]$ is a symmetric polynomial satisfying $\zeta(x) = p(x)p(x^{-1})\zeta'(x)$. Let

$$\mathbf{e}_1^\dagger(x) := \mathbf{u}^\dagger(x)\mathbf{P}'_1(x) \ . \tag{A.18}$$

The vector inside brackets in (A.17) must be self orthogonal since, otherwise, a degree-one building block can be extracted from the right of $\mathbf{F}_1(x, y)$. By the self-orthogonality property of this vector, we get

$$\Big[\gamma_0(x)\mathbf{v}^\dagger(x) + \zeta'(x)\mathbf{e}_1^\dagger(x)\Big]\Big[\overline{\gamma}_0(x)\mathbf{v}(x) + \zeta'(x)\mathbf{e}_1(x)\Big] = 0 \ .$$

Since $\mathbf{v}(x)$ is self orthogonal, this equation simplifies to

$$\gamma_0(x)\mathbf{v}^\dagger(x)\mathbf{e}_1(x) + \overline{\gamma_0}(x)\mathbf{e}_1^\dagger(x)\mathbf{v}(x) = \zeta'(x)\mathbf{e}_1^\dagger(x)\mathbf{e}_1(x) \ . \tag{A.19}$$

This equation holds for every symmetric polynomial $\zeta'(x)$. Since the left hand side is independent of $\zeta'(x)$, we must have $\mathbf{e}_1^\dagger(x)\mathbf{e}_1(x)$ that implies $\mathbf{e}_1(x)$ must be a self-orthogonal vector. Consequently, (A.19) further simplifies to

$$\gamma_0(x)\mathbf{v}^\dagger(x)\mathbf{e}_1(x) = \overline{\gamma_0}(x)\mathbf{e}_1^\dagger(x)\mathbf{v}(x) \ . \tag{A.20}$$

This equation holds for all symmetric polynomials $\gamma_0(x)$ without any condition on $\mathbf{e}_1(x)$. However, by its definition in (A.15), $\gamma_0(x)$ is determined by the choice for $\mu(x)$. Since we can always choose $\mu(x)$ in such a way that $\gamma_0(x)$ is nonsymmetric, for (A.20) to hold for all $\gamma_0(x)$, we must have $\mathbf{e}_1(x) = \mathbf{0}$. By this result and (A.18), we conclude that $\mathbf{u}(x) \in \mathsf{Null}_L(\mathbf{P}'_1(x))$, which means $\mathbf{P}'_1(x)$ is a singular polynomial matrix. Therefore, there exist a polynomial $\gamma_1(x) \in \mathbb{F}[x^{\pm 1}]$ and a polynomial vector $\boldsymbol{\theta}_1(x) \in (\mathbb{F}[x^{\pm 1}])^2$ such that

$$\mathbf{P}'_1(x) = \gamma_1(x)\,\mathbf{u}(x)\,\boldsymbol{\theta}_1^\dagger(x) \ . \tag{A.21}$$

Repeating a similar argument for $\mathsf{G}_2(x)$, defined in (A.16b), one can show that

$$\mathbf{P}'_1(x) = \gamma_1(x)\,\boldsymbol{\theta}_2(x)\,\mathbf{v}^\dagger(x) \ , \tag{A.22}$$

where $\boldsymbol{\theta}_2(x) \in (\mathbb{F}[x^{\pm 1}])^2$. Combining (A.21) and (A.22), we get $\boldsymbol{\theta}_1(x) = \mathbf{v}(x)$ and $\boldsymbol{\theta}_2(x) = \mathbf{u}(x)$, i.e.,

$$\mathbf{P}'_1(x) = \gamma_1(x)\,\mathbf{u}(x)\,\mathbf{v}^\dagger(x) \ .$$

Now, we set $\tau = 2$ and examine the possibility of factoring $\mathsf{S}_4\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big)$ from $\mu(x)\,\mathbf{P}(x,y)$. Let

$$\begin{aligned}\mathbf{F}'_1(x,y) &:= y^{-1}\mathsf{S}_4\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big)\ \mu(x)\mathbf{P}(x,y) \\ &= \underbrace{\Big[\gamma_0(x)\mathbf{A}^\dagger(x)\mathbf{u}(x)\mathbf{v}^\dagger(x) + \zeta(x)\mathbf{v}(x)\mathbf{u}^\dagger(x)\mathbf{P}'_2(x)\Big]}_{\mathsf{G}'_1(x)}\ y + \sum_{i=0}^{L}\mathbf{F}'_{1,i}(x)\,y^{-i}\end{aligned} \tag{A.23a}$$

and

$$\begin{aligned}\mathbf{F}'_2(x,y) &:= y^{-1}\mu(x)\mathbf{P}(x,y)\ \mathsf{S}_4\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big) \\ &= \underbrace{\Big[\gamma_0(x)\mathbf{u}(x)\mathbf{v}^\dagger(x)\mathbf{A}^\dagger(x) + \zeta(x)\mathbf{P}'_2(x)\mathbf{v}(x)\mathbf{u}^\dagger(x)\Big]}_{\mathsf{G}'_2(x)}\ y + \sum_{i=0}^{L}\mathbf{F}'_{2,i}(x)\,y^{-i} \ .\end{aligned} \tag{A.23b}$$

If either $\mathsf{G}'_1(x) = \mathbf{0}$ or $\mathsf{G}'_2(x) = \mathbf{0}$ for some $\zeta(x)$, we have successfully extracted a degree-four building block from $\mu(x)\mathbf{P}(x,y)$. However, since we have assumed $\mu(x)\mathbf{P}(x,y)$ is not factorable, none of $\mathsf{G}'_1(x)$ and $\mathsf{G}'_1(x)$ can be zero. Using an argument similar the one used for $\mathbf{P}'_1(x)$, we can show that

$$\mathbf{P}'_2(x,y) = \gamma_2(x)\,\mathbf{u}(x)\,\mathbf{v}^\dagger(x) \tag{A.24}$$

for some $\gamma_2(x) \in \mathbb{F}[x^{\pm 1}]$. Continuing the same process and increasing τ one at a time, we can show that

$$\mathbf{P}'_i(x,y) = \gamma_i(x)\,\mathbf{u}(x)\,\mathbf{v}^\dagger(x) \qquad \forall i \in [\,L\,] \ , \tag{A.25}$$

where $\gamma_i(x) \in \mathbb{F}[x^{\pm 1}]$ for all $i \in [\,L\,]$. An immediate implication of this result is that $\mathbf{P}^\dagger(x,y)\,\mathbf{P}(x,y) = \mathbf{0}$ that contradicts the assumption that $\mathbf{P}(x,y)$ is PU. Therefore, there exists an integer $\tau \in [\,L\,]$ such that either

$$\mathbf{P}'(x,y) = y^{-(\tau-1)}\,\mathsf{S}^\dagger_{2\tau}\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big)\,\mathbf{P}(x,y) \tag{A.26a}$$

or

$$\mathbf{P}'(x,y) = y^{-(\tau-1)}\,\mathbf{P}(x,y)\,\mathbf{S}^{\dagger}_{2\tau}\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big) \tag{A.26b}$$

is a polynomial matrix in $\mathsf{M}_2(\mathbb{F}[x^{\pm 1},y^{-1}])$ that is PU over $\mathbb{F}[x^{\pm 1}]$. After multiplying $\mathbf{P}'(x,y)$ by $\mathbf{S}_{2\tau}\Big(y;\,\mathbf{u}(x),\mathbf{v}(x),\zeta(x)\Big)$ (from left or right) and also $y^{\tau-1}$, we get the desired factorization.

To examine the possibility of degree reduction, we take the determinant of $\mathbf{P}'(x,y)$. By Lemma 3.3.3, we have

$$\det \mathbf{P}'(x,y) = [\det \mathbf{P}(x,y)]\; x^{m-r}\,p(x)\,p(x^{-1})\,q(x)\,q(x^{-1})\,y^{-2\tau}\;\;, \tag{A.27}$$

which implies the degree of $\mathbf{P}(x,y)$ with respect to y is reduced by 2τ. ∎

APPENDIX B

Efficient Generation of Multivariate PU Matrices

In this appendix, we provide efficient algorithms for generating univariate PU matrices in $\mathsf{PU}_n(\mathbb{F}[z^{-1}])$ or multivariate ones in $\mathsf{PU}_n(\mathbb{F}[z_1,\ldots,z_r])$, where $n \in \mathbb{Z}_{>2}$, $r \leq n$, and $\mathbb{F}$ is a field of characteristic two. As discussed before, PU matrices over the ring of polynomials with nonnegative and non-positive exponents are used to design asymmetric cryptosystems and causal wavelets, respectively. In fact, the signs of exponents are irrelevant to algorithms generating the building blocks. For convenience, we consider nonnegative exponents in this appendix.

The first step in generating PU matrices is generating univariate building blocks defined in (2.9), (2.10), and (2.12). To enhance further references, we have listed these building blocks in Table B.1. In the rest of this appendix, we introduce algorithms for generating the univariate building blocks and eventually explain how to efficiently multiply them. Outputs of the provided algorithms are in a special format suitable for the efficient multiplication.

Table B.1: PU building blocks and their parameters

Building Block	*Parameters*
$\mathbf{B}_1(z;\,\mathbf{v}) = \mathbf{I} + \mathbf{v}\mathbf{v}^\dagger + \mathbf{v}\mathbf{v}^\dagger z$	$\mathbf{v} \in \mathbb{F}^n, \quad \|\mathbf{v}\| = 1$
$\mathbf{B}_2(z;\,\mathbf{u},\mathbf{v}) = \mathbf{I} + \mathbf{u}\mathbf{v}^\dagger + \mathbf{v}\mathbf{u}^\dagger + (\mathbf{u}\mathbf{v}^\dagger + \mathbf{v}\mathbf{u}^\dagger)z$	$\mathbf{u},\mathbf{v} \in \mathbb{F}^n, \quad \|\mathbf{u}\| = \|\mathbf{v}\| = 0, \quad \mathbf{u}^\dagger\mathbf{v} = 1$
$\mathbf{R}_{n\tau}(z;\,\mathbf{V},\mathbf{\Lambda}) = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\dagger + \mathbf{I}z^\tau + \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\dagger z^{2\tau}$	$\tau \in \mathbb{N}, \quad \mathbf{\Lambda} = \mathsf{diag}(\lambda_1,\ldots,\lambda_n) \in \mathsf{M}_n(\mathbb{F})$
	$\mathbf{V} = [\,\mathbf{v}_1,\ldots,\mathbf{v}_n\,] \in \mathsf{M}_n(\mathbb{F}), \quad \mathbf{v}_i^\dagger\mathbf{v}_j = 0 \quad \forall i,j \in [\,n\,]$

1 Univariate Degree-One Building Block over $\mathbb{F}$

Since the characteristic of $\mathbb{F}$ is two, the norm of any vector $\mathbf{v} = [\, v_1, \ldots, v_n \,]^\dagger \in \mathbb{F}$ can be written as

$$\|\mathbf{v}\| = \sum_{i=1}^{n} v_i^2 = \left(\sum_{i=1}^{n} v_i \right)^2 \quad . \tag{B.1}$$

Hence, if $\mathbf{v}$ has a unit norm, its components satisfy

$$v_n = 1 + v_1 + \cdots + v_{n-1} \quad . \tag{B.2}$$

Using this observation, Algorithm B.1 can be used to generate such vectors. Since no multiplication is involved in this algorithm, it has constant time-complexity.

2 Univariate Degree-Two Building Block over $\mathbb{F}$

By Table B.1, two vectors $\mathbf{u} = [\, u_1, \ldots, u_n \,]^\dagger$ and $\mathbf{v} = [\, v_1, \ldots, v_n \,]^\dagger$ are required that satisfy the conditions $\|\mathbf{u}\| = \|\mathbf{v}\| = 0$ and $\mathbf{u}^\dagger \mathbf{v} = 1$. These conditions impose the three equations

$$u_1 + \cdots + u_n = 0 \ , \qquad v_1 + \cdots + v_n = 0 \ , \qquad u_1\, v_1 + \cdots + u_n\, v_n = 1 \tag{B.3}$$

while we have $2n$ unknowns. Therefore, the degree of freedom is $2n - 3$. Without loss of generality, we may assume only u_n, v_{n-1}, and v_n are unknown and the rest of the vector components are given. Since $\mathbf{u}$ is self orthogonal, we have $\sum_{i=1}^{n} u_i = 0$ that yields $u_n = \sum_{i=1}^{n-1} u_i$. By the self orthogonality of $\mathbf{v}$ and $\mathbf{u}^\dagger \mathbf{v} = 1$, we obtain the following system of linear equations

$$\begin{cases} v_{n-1} + v_n = \alpha \\ u_{n-1} v_{n-1} + u_n v_n = \beta \end{cases} \tag{B.4}$$

Algorithm B.1: Degree-One Building Block Generation

Input: Vector $\hat{\mathbf{v}} = [\, v_1, \ldots, v_{n-1} \,]^\dagger \in \mathbb{F}^{n-1}$

Output: Vector $\mathbf{v} \in \mathbb{F}^n$ such that $\mathbf{v}^\dagger \mathbf{v} = 1$

1. $\mathbf{v} \leftarrow \left[\hat{\mathbf{v}}^\dagger, 1 + v_1 + \cdots + v_{n-1} \right]^\dagger$

---

in the unknowns v_{n-1} and v_n, where $\alpha = \sum_{i=1}^{n-2} v_i$ and $\beta = 1 + \sum_{i=1}^{n-2} u_i\, v_i$. The solution to this system is

$$v_{n-1} = \frac{\beta + \alpha\, u_n}{u_{n-1} + u_n} \ , \qquad v_n = \frac{\beta + \alpha\, u_{n-1}}{u_{n-1} + u_n} \ . \tag{B.5}$$

Using these results, Algorithm B.2 is suggested for designing parameters of the degree-two building block. In this algorithm, calculations of β and v_{n-1} require $n-2$ and 2 multiplications, respectively. Therefore, the time complexity of Algorithm B.2 is $O(n)$.

3 Univariate Degree-$n\tau$ Building Block over $\mathbb{F}$

By Table B.1, the generation of the degree-$n\tau$ building block takes a set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ such that $\mathbf{v}_i^\dagger \mathbf{v}_j = 0$ for all $i, j \in [\, n\,]$. These conditions impose $n(n+1)/2$ equations in n^2 unknowns, which yields the degree of freedom $n(n-1)/2$. Therefore, without loss of generality, we assume the last $n-i$ components of the vector $\mathbf{v}_i$ are fixed and given for all $i \in [\, n\,]$. Let $\mathbf{v}_i = [\, v_{i,1}, \ldots, v_{i,n}\,]^\dagger$ for all $i \in [\, n\,]$. To solve the system of equations for unknowns, we start with $\mathbf{v}_1$. This vector has a single unknown $v_{1,1}$ that is obtained as $v_{1,1} = \sum_{i=2}^{n} v_{1,i}$ using $\|\mathbf{v}_1\| = 0$. Proceeding toward $\mathbf{v}_n$, we completely calculate all the unknown components of vectors one at a time. At the k-th step, we have completely

---

Algorithm B.2: Degree-Two Building Block Generation

Input: Vectors $\hat{\mathbf{u}} = [\, u_1, \ldots, u_{n-1}\,]^\dagger \in \mathbb{F}^{n-1}$ and $\hat{\mathbf{v}} = [\, v_1, \ldots, v_{n-2}\,]^\dagger \in \mathbb{F}^{n-2}$

Output: Vectors $\mathbf{u} \in \mathbb{F}^n$ and $\mathbf{v} \in \mathbb{F}^n$ such that $\|\mathbf{u}\| = \|\mathbf{v}\| = 0$ and $\mathbf{u}^\dagger \mathbf{v} = 1$

1. $u_n \leftarrow u_1 + \cdots + u_{n-1}$
2. $\alpha \leftarrow v_1 + \cdots + v_{n-2}$
3. $\beta \leftarrow 1 + u_1 v_1 + \cdots + u_{n-2} v_{n-2}$
4. $v_{n-1} \leftarrow (\beta + \alpha\, u_n)/(u_{n-1} + u_n)$
5. $v_n \leftarrow \alpha + v_{n-1}$
6. $\mathbf{u} \leftarrow \big[\, \hat{\mathbf{u}}^\dagger,\ u_n\,\big]^\dagger$
7. $\mathbf{v} \leftarrow \big[\, \hat{\mathbf{v}}^\dagger,\ v_{n-1},\ v_n\,\big]^\dagger$

---

calculated all the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}$. The vector $\mathbf{v}_k$ has k unknowns $v_{k,1}, \ldots, v_{k,k}$ that satisfy the following linear equations

$$\begin{cases} v_{k,1} + \cdots + v_{k,k} = \alpha_{k,0} \\ v_{1,1}\, v_{k,1} + \ldots + v_{1,k}\, v_{k,k} = \alpha_{k,1} \\ \qquad\qquad \vdots \\ v_{k-1,1}\, v_{k,1} + \cdots + v_{k-1,k}\, v_{k,k} = \alpha_{k,k-1} \ , \end{cases} \tag{B.6}$$

where $\alpha_{k,0} = \sum_{i=k+1}^{n} v_{k,i}$ and $\alpha_{k,j} = \sum_{i=k+1}^{n} v_{j,i}\, v_{k,i}$ for all $j \in [\, k-1 \,]$. The complexity of solving this system using Gaussian elimination is $O(k^3)$, where $k = 2, 3, \ldots, n$. Therefore, the total complexity of calculating the matrix $\mathbf{V}$ is $O(n^4)$. Combining these results, Algorithm B.3 is used to generate the degree-$n\tau$ building block.

Algorithm B.3: Degree-$n\tau$ Building Block Generation

Input: Vectors $\hat{\mathbf{v}}_1 \in \mathbb{F}^{n-1}$, $\hat{\mathbf{v}}_2 \in \mathbb{F}^{n-2}$, $\ldots$, $\hat{\mathbf{v}}_{n-1} \in \mathbb{F}$

Output: Vectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\ldots$, $\mathbf{v}_n \in \mathbb{F}^n$ such that $\mathbf{v}_i^{\dagger}\, \mathbf{v}_j = 0$ for all $i, j \in [\, n \,]$

▷ Assume $\hat{\mathbf{v}}_i = [\, v_{i,i+1},\, \ldots,\, v_{i,n} \,]^{\dagger}$.

1. $v_{1,1} \leftarrow \sum_{i=2}^{n} v_{1,i}$
2. $\mathbf{v}_1 \leftarrow \left[\, v_{1,1},\, \hat{\mathbf{v}}_1^{\dagger} \,\right]^{\dagger}$
3. **for** $k = 2$ **to** $n-1$ **do**
4. Solve the linear system of equations B.6 for $v_{k,1}, \ldots, v_{k,k}$ using Gaussian elimination.
5. $\mathbf{v}_k \leftarrow \left[\, v_{k,1},\, \ldots,\, v_{k,k},\, \hat{\mathbf{v}}_k^{\dagger} \,\right]^{\dagger}$
6. **end**
7. Solve the linear system B.6 for $v_{n,1}, \ldots, v_{n,n}$ using Gaussian elimination.
8. $\mathbf{v}_n \leftarrow [\, v_{n,1},\, \ldots,\, v_{n,n} \,]^{\dagger}$

---

4 Efficient Multiplication of PU Building Blocks

For given and fixed integers $r, N \in \mathbb{N}$, assume the goal is calculating an r-variate PU matrix $\mathbf{P}(\mathbf{z})$ consisting of N univariate building blocks in each variable, i.e.,

$$\mathbf{P}(\mathbf{z}) = \prod_{i=1}^{rN} \mathbf{C}_{\sigma(i)}(z_{\lceil \sigma(i)/N \rceil}) \ , \tag{B.7}$$

where $\sigma \in \mathfrak{S}_{rN}$ is a fixed permutation and $\mathbf{C}_i(z)$ is one of the univariate building blocks in Table B.1. The special structure of these building blocks makes their multiplication less complex than multiplying arbitrary matrices. By induction, it can be easily shown that these building blocks and their multiplications have the following form

$$\mathbf{C}(\mathbf{z}) = \mathbf{I}\, \mathbf{z}^{\boldsymbol{\beta}} + \sum_{\boldsymbol{\alpha} \in \mathcal{A}} \sum_{j \in \mathcal{J}} \mathbf{u}_{\boldsymbol{\alpha},j}\, \mathbf{v}_{\boldsymbol{\alpha},j}^{\dagger}\, \mathbf{z}^{\boldsymbol{\alpha}} \ , \tag{B.8}$$

where $\mathbf{u}_{\boldsymbol{\alpha}j}, \mathbf{v}_{\boldsymbol{\alpha}j} \in \mathbb{F}^n$, $\boldsymbol{\beta} \in \mathbb{Z}^r_{\geq 0}$, $\mathcal{A} \subset \mathbb{Z}^r_{\geq 0}$, and $\mathcal{J} \subset \mathbb{N}$ such that $\mathcal{A}$ and $\mathcal{J}$ are finite sets. We notice that the matrix $\mathbf{C}$ is completely determined if the exponent vector $\boldsymbol{\beta}$ and the sets $\mathcal{A}$ and $\mathcal{J}$ along with the following sets of vectors are known.

$$\mathcal{U}(\mathbf{C}) := \{\, \mathbf{u}_{\boldsymbol{\alpha},j} \,:\, \boldsymbol{\alpha} \in \mathcal{A}, j \in \mathcal{J} \,\} \tag{B.9a}$$

$$\mathcal{V}(\mathbf{C}) := \{\, \mathbf{v}_{\boldsymbol{\alpha},j} \,:\, \boldsymbol{\alpha} \in \mathcal{A}, j \in \mathcal{J} \,\} \tag{B.9b}$$

Hence, if $\mathbf{D}$ is one of the intermediate matrices in the process of multiplying the PU matrices $\mathbf{D}_1, \ldots, \mathbf{D}_L$, instead of multiplying the vectors $\mathbf{u}_{\boldsymbol{\alpha}j}$ and $\mathbf{v}_{\boldsymbol{\alpha}j}$, the sets $\mathcal{U}(\mathbf{D})$ and $\mathcal{V}(\mathbf{D})$ are obtained. That is why the generating algorithms for the building blocks only compute the vector parameters of these building blocks. The advantage of this strategy is reducing the complexity of multiplying matrices. To find the complexity of matrix multiplication using this method, we provide some lemmas.

LEMMA B.1. *Let* $\mathbf{T}_{\mathbf{a}}(\mathbf{z};\, \mathbf{u}_{\mathbf{a}}, \mathbf{v}_{\mathbf{a}}) := \mathbf{u}_{\mathbf{a}}\, \mathbf{v}_{\mathbf{a}}^{\dagger}\, \mathbf{z}^{\mathbf{a}} \in \mathsf{M}_n(\mathbb{F}[z])$, *where* $\mathbf{u}_{\mathbf{a}}, \mathbf{v}_{\mathbf{a}} \in \mathbb{F}^n$ *and* $\mathbf{a} \in \mathbb{Z}^r_{\geq 0}$. *For fixed* $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}^r_{\geq 0}$, *the total complexity of calculating* $\mathcal{U}(\mathbf{T}_{\boldsymbol{\alpha}}\, \mathbf{T}_{\boldsymbol{\beta}})$ *and* $\mathcal{V}(\mathbf{T}_{\boldsymbol{\alpha}}\, \mathbf{T}_{\boldsymbol{\beta}})$ *is* $O(n)$.

Proof. By direct multiplication, we have

$$\mathbf{T}_{\boldsymbol{\alpha}}(\mathbf{z};\, \mathbf{u}_{\boldsymbol{\alpha}}, \mathbf{v}_{\boldsymbol{\alpha}})\, \mathbf{T}_{\boldsymbol{\beta}}(\mathbf{z};\, \mathbf{u}_{\boldsymbol{\beta}}, \mathbf{v}_{\boldsymbol{\beta}}) = (\mathbf{v}_{\boldsymbol{\alpha}}^{\dagger}\, \mathbf{u}_{\boldsymbol{\beta}})\, \mathbf{u}_{\boldsymbol{\alpha}}\, \mathbf{v}_{\beta}^{\dagger}\, \mathbf{z}^{\boldsymbol{\alpha}+\boldsymbol{\beta}} \ .$$

The constant scalar $\mathbf{v}_{\boldsymbol{\alpha}}^{\dagger}\,\mathbf{u}_{\boldsymbol{\beta}}$ can be absorbed either by $\mathbf{u}_{\boldsymbol{\alpha}}$ or by $\mathbf{v}_{\boldsymbol{\beta}}$. Assuming it is absorbed by the first one, we have $\mathcal{U}(\mathbf{T}_{\boldsymbol{\alpha}}\,\mathbf{T}_{\boldsymbol{\beta}}) = \left\{\,(\mathbf{v}_{\boldsymbol{\alpha}}^{\dagger}\,\mathbf{u}_{\boldsymbol{\beta}})\,\mathbf{u}_{\boldsymbol{\alpha}}\,\right\}$ and $\mathcal{U}(\mathbf{T}_{\boldsymbol{\alpha}}\,\mathbf{T}_{\boldsymbol{\beta}}) = \{\,\mathbf{v}_{\beta}\,\}$. The complexity of calculating the first set is $2n$ while the other one obtains for free. Hence, the total complexity is $O(n)$. ∎

LEMMA B.2. *Let $\mathbf{D}_1, \cdots, \mathbf{D}_L$ be matrices each of the form (B.8) for some $L \in \mathbb{N}$. Then, the complexity of computing both $\mathcal{U}(\prod_{i=1}^{L}\mathbf{D}_i)$ and $\mathcal{V}(\prod_{i=1}^{L}\mathbf{D}_i)$ is $O(n\,\prod_{i=1}^{L}c_i)$, where $c_i = 1 + |\mathcal{U}(\mathbf{D}_i)|$ is the number of terms of $\mathbf{D}_i$ for all $i \in [\,L\,]$.*

Proof. Let

$$\mathbf{D}_i(\mathbf{z}) = \mathbf{I}\,\mathbf{z}^{\boldsymbol{\beta}_i} + \sum_{\boldsymbol{\alpha}\in\mathcal{A}_i}\sum_{j\in\mathcal{J}_i}\mathbf{u}_{i,\boldsymbol{\alpha},j}\,\mathbf{v}_{i,\boldsymbol{\alpha},j}^{\dagger}\,\mathbf{z}^{\boldsymbol{\alpha}}\ .$$

By the definition of the matrix $\mathbf{T}$ in Lemma B.1, we have

$$\mathbf{D}_i(\mathbf{z}) = \mathbf{I}\,\mathbf{z}^{\boldsymbol{\beta}_i} + \sum_{\boldsymbol{\alpha}\in\mathcal{A}_i}\sum_{j\in\mathcal{J}_i}\mathbf{T}_{\boldsymbol{\alpha}}(\mathbf{z};\,\mathbf{u}_{i,\boldsymbol{\alpha},j},\mathbf{v}_{i,\boldsymbol{\alpha},j})\ .$$

Therefore, by Lemma B.1, the complexity of calculating both sets $\mathcal{U}(\prod_{i=1}^{L}\mathbf{D}_i)$ and $\mathcal{V}(\prod_{i=1}^{L}\mathbf{D}_i)$ is approximately $O(n\,\prod_{i=1}^{L}c_i)$. ∎

Using the suggested technique, to obtain the matrix $\mathbf{P}(\mathbf{z})$ in (B.7), we first calculate the sets $\mathcal{U}(\mathbf{P})$ and $\mathcal{V}(\mathbf{P})$. We can state the following fact about the matrix $\mathbf{P}$ using Lemma B.2 and by the fact that the degree-one, degree-two, and degree-$n\tau$ building blocks have three, five, and $2n+1$ terms, respectively.

FACT B.1. *Assume the PU matrix $\mathbf{P}$ in (B.7) is constructed by N building blocks in the variable z_i among which $c_i^d \in \{\,0,1,\ldots,N\,\}$ of them are degree-d building blocks for $d \in \{\,1,2,n\tau\,\}$, where $c_i^1 + c_i^2 + c_i^{n\tau} = N$, for all $i \in [\,r\,]$. Then,*

(i) The total number of terms in $\mathbf{P}$ is upper bounded by

$$c \le \left(3^{\sum_{i=1}^{r}c_i^1}\right)\left(5^{\sum_{i=1}^{r}c_i^2}\right)\left((2n+1)^{\sum_{i=1}^{r}c_i^{n\tau}}\right) = O\left(n^{\sum_{i=1}^{r}c_i^{n\tau}}\right)\ . \tag{B.10}$$

(ii) The total complexity of calculating the sets $\mathcal{U}(\mathbf{P})$ and $\mathcal{V}(\mathbf{P})$ is

$$O(nc) = O\left(n^{1+\sum_{i=1}^{r}c_i^{n\tau}}\right)\ . \tag{B.11}$$

An immediate consequence of this fact is the following result.

FACT B.2. *If we employ only the degree-one and degree-two building blocks in the construction of the PU matrix* $\mathbf{P}$ *in (B.7), then*

(i) The number of terms of the matrix polynomial $\mathbf{P}(z)$ *is independent of* n*, i.e.,* $|\mathcal{U}(\mathbf{P})| = O(1)$.

(ii) The total complexity of calculating the sets $\mathcal{U}(\mathbf{P})$ *and* $\mathcal{V}(\mathbf{P})$ *is* $O(n)$.

Having the two sets $\mathcal{U}(\mathbf{P})$ and $\mathcal{V}(\mathbf{P})$, the final step is constructing the matrix $\mathbf{P}(z)$ that consists of the summation of terms of the form $\mathbf{u}\,\mathbf{v}^\dagger$, where $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$. This construction is performed by carrying out the multiplication $\mathbf{u}\,\mathbf{v}^\dagger$, which has complexity n, for every term. Hence, assuming that only the degree-one and degree-two building blocks are employed, we have the following fact.

FACT B.3. *Having the sets* $\mathcal{U}(\mathbf{P})$ *and* $\mathcal{V}(\mathbf{P})$*, the total complexity of constructing* $\mathbf{P}(z)$ *is* $|\mathcal{U}(\mathbf{P})|\ n^2 = O(n^2)$.

APPENDIX C

Toy Examples of the PAC and PDSS

In this appendix, we provide toy examples for both the public-key cryptosystem PAC and the signature scheme PDSS introduced in Chapter 6. We note that these are not practical examples, and the resulting schemes are insecure in practice because of the small choices for parameters. The purpose of these examples is to show how the introduced systems are designed and illustrate the structure of public polynomials. In our design, we have used the computer algebra software SINGULAR [99].

In both examples, we assume $n = 3$ and $r = 1$, the operating field is $\mathsf{GF}(256)$, and ε is a primitive element of the field.

1 A Toy Example of PAC

In our design, we follow the guidelines of Section 6.5. Since $r = 1$, the PU matrix $\mathbf{P}(z)$ is a univariate polynomial matrix. We construct it with one degree-one building block as in (2.9) with the unit-norm vector $\mathbf{v} = [\,\varepsilon,\, \varepsilon^5,\, \varepsilon^{47}\,]^\dagger$. The resulting PU matrix is

$$\mathbf{P}(z) = \begin{bmatrix} \varepsilon^{50} + \varepsilon^2 z & \varepsilon^6 + \varepsilon^6 z & \varepsilon^{48} + \varepsilon^{48} z \\ \varepsilon^6 + \varepsilon^6 z & \varepsilon^{21} + \varepsilon^{10} z & \varepsilon^{52} + \varepsilon^{52} z \\ \varepsilon^{48} + \varepsilon^{48} z & \varepsilon^{52} + \varepsilon^{52} z & \varepsilon^{202} + \varepsilon^{94} z \end{bmatrix} . \tag{C.1}$$

As suggested in (6.43) and (6.44), we use the tame automorphisms

$$\mathbf{t}_1(\mathbf{x}) = \begin{bmatrix} x_1 + \varepsilon^3 \\ x_2 + \varepsilon^4 x_1^3 + 1 \\ x_3 + x_1^5 x_2^6 + \varepsilon^{14} \end{bmatrix} \qquad \mathbf{t}_2(\mathbf{x}) = \begin{bmatrix} x_3 \\ x_2 + \varepsilon^{11} x_3^2 \\ x_1 + \varepsilon^3 x_2^2 x_3 \end{bmatrix} . \tag{C.2}$$

The composite tame automorphism $\mathbf{t} = \mathbf{t}_1 \circ \mathbf{t}_2 = [\, t_1,\, t_2,\, t_3 \,]^\dagger$ is

$$\begin{aligned} t_1(\mathbf{x}) &= \varepsilon^{14} + x_3 + x_1^5 x_2^6 \\ t_2(\mathbf{x}) &= \varepsilon^{106} + x_2 + \varepsilon^{11} x_3^2 + \varepsilon^4 x_1^3 + \varepsilon^{11} x_1^{10} x_2^{12} \\ t_3(\mathbf{x}) &= \varepsilon^{227} + x_1 + \varepsilon^3 x_3 + \varepsilon^{17} x_2^2 + \varepsilon^3 x_2^2 x_3 + \varepsilon^{25} x_1^6 \\ &\quad + \varepsilon^{11} x_1^6 x_3 + \varepsilon^3 x_1^5 x_2^6 + \varepsilon^3 x_1^5 x_2^8 + \varepsilon^{11} x_1^{11} x_2^6 \ . \end{aligned} \tag{C.3}$$

The vector polynomial $\boldsymbol{\rho}$ in (6.40) is a one-dimensional multivariate polynomial. We choose its coefficients and exponents as follows.

$$\rho(\mathbf{x}) = \varepsilon x_1 + \varepsilon^3 x_2^8 x_3^6 \tag{C.4}$$

For the irreducible polynomial $\chi(x)$ in (6.36), we use $\omega = \varepsilon^5$ since $\mathrm{Tr}(\varepsilon^5) \neq 0$. These choices give the following irreducible multivariate polynomial for the vector polynomial $\boldsymbol{\varphi}(\mathbf{x})$ in (6.34).

$$\varphi(\mathbf{x}) = \varepsilon^5 + \varepsilon x_1 + \varepsilon^2 x_1^2 + \varepsilon^3 x_2^8 x_3^6 + \varepsilon^6 x_2^{16} x_3^{12} \tag{C.5}$$

In the design of the unitary matrix $\mathbf{A}$ in (6.27), we use the building block $\mathbf{U}_{\zeta,\mathbf{v}}$ in (2.7) with $\zeta = 1$ and the self-orthogonal vector $\mathbf{v} = \big[\, 1,\, \varepsilon,\, \varepsilon^6,\, \varepsilon^{98} \,\big]^\dagger$. The resulting unitary matrix is

$$\mathbf{A} = \begin{bmatrix} 0 & \varepsilon & \varepsilon^6 & \varepsilon^{98} \\ \varepsilon & \varepsilon^{50} & \varepsilon^7 & \varepsilon^{99} \\ \varepsilon^6 & \varepsilon^7 & \varepsilon^{127} & \varepsilon^{104} \\ \varepsilon^{98} & \varepsilon^{99} & \varepsilon^{104} & \varepsilon^{23} \end{bmatrix} \ . \tag{C.6}$$

The constant vector $\mathbf{b}$ in (6.27) is chosen to be $\mathbf{b} = \big[\, \varepsilon^3,\, \varepsilon^2,\, 1,\, \varepsilon^{17} \,\big]^\dagger$.

As stated in Fact 6.5.6, the entries of the OWF $\boldsymbol{\Psi}$ are polynomials whose monomials are subsets of a maximal set of monomials. Hence, we just give one of the public polynomials. The rest of them have similar structures. If $\boldsymbol{\Psi}(\mathbf{x}) = [\, \Psi_1(\mathbf{x}),\, \Psi_2(\mathbf{x}),\, \Psi_3(\mathbf{x}) \,]^\dagger$, then the terms of the polynomial $\Psi_1(\mathbf{x})$ are given in Table C.1. The row $a, \alpha_1, \alpha_2, \alpha_3$ in this table corresponds to the term $\varepsilon^a\, x_1^{\alpha_1}\, x_2^{\alpha_2}\, x_3^{\alpha_3}$ in the polynomial.

The SINGULAR code used to generate this example is provided in the following.

Table C.1: Coefficients of the polynomial $\Psi_1(\mathbf{x})$ in PAC

ε	x_1	x_2	x_3	ε	x_1	x_2	x_3	ε	x_1	x_2	x_3	ε	x_1	x_2	x_3
47	0	0	0	158	1	2	1	157	1	8	6	101	0	16	12
82	1	0	0	125	2	0	2	115	0	9	6	160	1	16	12
186	0	1	0	118	5	0	0	53	0	8	7	118	0	17	12
46	0	0	1	159	2	2	1	174	0	10	6	56	0	16	13
204	2	0	0	76	6	0	0	126	0	8	8	177	0	18	12
113	1	1	0	180	7	0	0	62	11	6	0	129	0	16	14
68	0	2	0	62	6	0	1	119	3	8	6	168	11	14	6
51	1	0	1	181	8	0	0	160	0	10	7	122	3	16	12
197	0	0	2	166	7	0	1	166	12	6	0	163	0	18	13
37	3	0	0	167	8	0	1	167	13	6	0	185	6	16	12
114	2	1	0	46	5	6	0	182	6	8	6	171	6	16	13
172	1	2	0	51	6	6	0	168	6	8	7	126	10	20	6
52	2	0	1	52	7	6	0	197	10	12	0	56	5	22	12
54	0	2	1	54	5	8	0	124	11	12	0	163	5	24	12
124	1	0	2	158	6	8	0	125	12	12	0	171	11	22	12
117	4	0	0	98	0	8	6	53	5	14	6	129	10	28	12
173	2	2	0	159	7	8	0	160	5	16	6				

```
1  // This is the toy example provided for the PAC
2  //
3  ring r = (256,a), (x(1..3)), ds;
4  // Use minpoly to specify the minimal polynomial
5  //
6  poly z = a*x(1) + a^3*x(2)^8*x(3)^6;
7  poly w = z^2 + z + a^5;
8  //
9  matrix I[3][3] = 1, 0, 0, 0, 1, 0, 0, 0, 1;
```

```
10  //
11  matrix c[2][1] = a, a^5;
12  matrix v[3][1] = c, 1+c[1,1]+c[2,1];
13  matrix P[3][3] = I + v*transpose(v) + v*transpose(v)*w;
14  //
15  matrix t1[3][1] = x(1) + a^3,
16                    x(2)+a^4*x(1)^3 + 1,
17                    x(3)+x(1)^5*x(2)^6 + a^14;
18  matrix t[3][1] = t1[3,1],
19                   t1[2,1] + a^11*t1[3,1]^2,
20                   t1[1,1] + a^3*t1[2,1]^2*t1[3,1];
21  //
22  matrix g1[3][1] = P*t;
23  //
24  matrix g2[4][1] = transpose(g1), w;
25  //
26  matrix I[4][4] = 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1;
27  matrix d[3][1] = 1, a, a^6;
28  matrix u[4][1] = d, d[1,1]+d[2,1]+d[3,1];
29  matrix A[4][4] = I + u*transpose(u);
30  //
31  matrix b[4][1] = a^3, a^2, 1, a^17;
32  //
33  matrix f[4][1] = A*g2+b;
```

2 A Toy Example of the PDSS

In the toy example of the PDSS, we use the same PU matrix as in (C.1) and the same automorphisms $\mathbf{t}_1$ and $\mathbf{t}_2$ as in (C.2). For the vector polynomial $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{x}')$, we use

$$\varphi(\mathbf{x}, x') = \varepsilon x' + \varepsilon^2 x_1 + \varepsilon^9 x_2^5\, x_3^7 \ . \tag{C.7}$$

Assuming $\mathbf{\Psi}(\mathbf{x}, x') = [\,\Psi_1(\mathbf{x}, x'),\, \Psi_2(\mathbf{x}, x'),\, \Psi_3(\mathbf{x}, x')\,]^{\dagger}$, the terms of the polynomial $\Psi_1(\mathbf{x}, x')$ are given in Table C.2. The row $a, \alpha_1, \alpha_2, \alpha_3, \alpha'$ in this table corresponds to the term $\varepsilon^a\, x_1^{\alpha_1}\, x_2^{\alpha_2}\, x_3^{\alpha_3}\, (x')^{\alpha'}$ in the polynomial.

The SINGULAR code used to generate this example is provided in the following.

```
1  // This is the toy example provided for the PDSS
2  //
3  ring r = (256,a), (x(1..4)), ds;
4  // Use minpoly to specify the minimal polynomial
5  //
6  poly z = a*x(4) + a^2*x(1) + a^9*x(2)^5*x(3)^7;
7  //
8  matrix I[3][3] = 1, 0, 0, 0, 1, 0, 0, 0, 1;
```

Table C.2: Coefficients of the polynomial $\Psi_1(\mathbf{x}, x')$ in PDSS

ε	x_1	x_2	x_3	x'	ε	x_1	x_2	x_3	x'	ε	x_1	x_2	x_3	x'	ε	x_1	x_2	x_3	x'
40	0	0	0	0	67	1	2	0	0	60	6	0	1	1	60	0	7	8	0
162	1	0	0	0	51	0	2	1	0	75	5	6	0	0	59	11	6	0	0
6	0	1	0	0	19	1	0	2	0	201	6	6	0	0	61	12	6	0	0
75	0	0	1	0	66	0	2	0	1	221	0	5	7	0	82	6	5	7	0
213	0	0	0	1	18	0	0	2	1	200	5	6	0	1	60	11	6	0	1
50	2	0	0	0	12	4	0	0	0	51	5	8	0	0	68	6	5	8	0
8	1	1	0	0	53	1	2	1	0	57	1	5	7	0	17	10	12	0	0
65	0	2	0	0	11	3	0	0	1	15	0	6	7	0	19	11	12	0	0
201	1	0	1	0	52	0	2	1	1	208	0	5	8	0	208	5	11	7	0
17	0	0	2	0	73	6	0	0	0	53	6	8	0	0	18	10	12	0	1
49	1	0	0	1	75	7	0	0	0	74	0	7	7	0	60	5	13	7	0
7	0	1	0	1	59	6	0	1	0	26	0	5	9	0	68	11	11	7	0
200	0	0	1	1	74	6	0	0	1	52	5	8	0	1	26	10	17	7	0
10	3	0	0	0	61	7	0	1	0	19	3	5	7	0					

```
 9  //
10  matrix c[2][1] = a, a^5;
11  matrix v[3][1] = c, 1+c[1,1]+c[2,1];
12  matrix P[3][3] = I + v*transpose(v) + v*transpose(v)*z;
13  //
14  matrix t1[3][1] = x(1) + a^3,
15                    x(2)+a^4*x(1)^3 + 1,
16                    x(3)+x(1)^5*x(2)^6 + a^14;
17  matrix t[3][1] = t1[3,1],
18                   t1[2,1] + a^11*t1[3,1]^2,
19                   t1[1,1] + a^3*t1[2,1]^2*t1[3,1];
20  //
21  matrix f[3][1] = P*t;
```

APPENDIX D

Details and Proofs of MKPS

1 ID-Assignment Algorithm

Let n be the maximum number of sensor nodes in the network, $m = \lceil \sqrt[d]{n} \rceil$, and $\mathcal{I}$ be the set of all IDs. Since $[\, m-1 \,]^d \subsetneq \mathcal{I} \subseteq [\, m \,]^d$, all the d tuples in $[\, m-1 \,]^d$ are assigned as node IDs. The remaining IDs come from the set $[\, m \,]^d \setminus [\, m-1 \,]^d$. This set can be partitioned as $[\, m \,]^d \setminus [\, m-1 \,]^d = \bigcup_{j=0}^{d-1} \mathcal{I}_j$, where

$$\mathcal{I}_j := \Big\{ (\, i_0, \ldots, i_{j-1}, m-1, i_{j+1}, \ldots, i_{d-1} \,) \; : \\ 0 \le i_0, \ldots, i_{j-1} \le m-2,\, 0 \le i_{j+1}, \ldots, i_{d-1} \le m-1 \Big\} \; . \quad \text{(D.1)}$$

The ID-assignment algorithm continues by assigning the vectors in $\mathcal{I}_j$ progressing from $j = 0$ to $j = d-1$. Since every $I \in \mathcal{I}_j$ is determined by $I\langle j \rangle$, the task of assigning the IDs in $\mathcal{I}_j$ in the d-dimensional space reduces to assigning the vector $I\langle j \rangle$ in the $(d-1)$-dimensional space. The ID-assignment algorithm can be recursively used since the dimension is reduced by one at each step. The recursion stops at dimension two in which case the grid-assignment algorithm of [127] is employed.

2 Average Probability of Link-Key Establishment

Given a node $I \in \mathcal{I}$, the probability of establishing a link key with this node is

$$P_{lk,I} = \frac{X_I}{n-1} \; , \quad \text{(D.2)}$$

where $X_I := \big|\{\, I' \in \mathcal{I} : h(I, I') = 1 \,\}\big|$. If n is a perfect dth root, then $\mathcal{I} = [\,m\,]^d$ and $X_I = d(m-1)$ for all $I \in \mathcal{I}$. Otherwise, $\mathcal{I} \subsetneq [\,m\,]^d$ is a proper subset, and $X_I \leq d(m-1)$. The average probability of link-key establishment is

$$P_{lk} = \frac{1}{n-1} E\left(X_I \,|\, I \in [\,m-1\,]^d\right) \mathsf{Prob}\left(I \in [\,m-1\,]^d\right) + \frac{1}{n-1} E\left(X_I \,|\, I \in \mathcal{I} \setminus [\,m-1\,]^d\right) \mathsf{Prob}\left(I \in \mathcal{I} \setminus [\,m-1\,]^d\right) \ , \quad \text{(D.3)}$$

where $\mathsf{Prob}\left(I \in [\,m-1\,]^d\right) = (m-1)^d/n$ and $\mathsf{Prob}\left(I \in \mathcal{I} \setminus [\,m-1\,]^d\right) = 1 - (m-1)^d/n$. For $I \in [\,m-1\,]^d$, we have

$$X_I = d(m-2) + \sum_{j=0}^{d-1} Y_{I,j} \ , \quad \text{(D.4)}$$

where

$$Y_{I,j} := \begin{cases} 1, & \exists I' \in \mathcal{I}_j \cap \mathcal{I} \text{ such that } h(I, I') = 1 \\ 0, & \text{otherwise} \ . \end{cases} \quad \text{(D.5)}$$

and the set $\mathcal{I}_j$ is defined in (D.1). For all $j \in [\,d\,]$, we define the set $\mathcal{N}_j$ as

$$\mathcal{N}_j := \Big\{\, (\,i_0, \ldots, i_{j-1}, m-1, i_{j+1}, \ldots, i_{d-1}\,) \in \mathcal{I}_j \cap \mathcal{I} : 0 \leq i_0, \ldots, i_{j-1}, i_{j+1}, \ldots, i_{d-1} \leq m-2 \Big\} \ . \quad \text{(D.6)}$$

It can be easily shown that

$$E\left(X_I \,|\, I \in [\,m-1\,]^d\right) = d(m-2) + \frac{\sum_{j=0}^{d-1} |\mathcal{N}_j|}{(m-1)^{d-1}} \ . \quad \text{(D.7)}$$

Let $0 \leq \nu \leq d-1$ be the maximum integer such that all the vectors in $\mathcal{I}_0, \ldots, \mathcal{I}_{\nu-1}$ are completely assigned using the ID-assignment algorithm. Hence, ν is the maximum integer such that $\sum_{j=0}^{\nu-1} |\mathcal{I}_j| \leq \left|\mathcal{I} \setminus [\,m-1\,]^d\right|$. Since $|\mathcal{I}_j| = (m-1)^j m^{d-j-1}$ for all $j \in [\,d\,]$, it is easy to show that

$$\nu = \left\lfloor \frac{\log\left(1 + \theta^d - nm^{-d}\right)}{\log \theta} \right\rfloor \ , \quad \text{(D.8)}$$

where

$$\theta := 1 - \frac{1}{m} \ . \quad \text{(D.9)}$$

Hence, we have $\sum_{j=0}^{d-1} |\mathcal{N}_j| \approx \nu(m-1)^{d-1}$. Substituting this result in (D.7), we get $E\Big(X_I \,|\, I \in [\,m-1\,]^d\Big) = d(m-2) + \nu$.

To compute $E\left(X_I \,|\, I \in \mathcal{I} \setminus [\,m-1\,]^d\right)$, we note that

$$E\left(X_I \,|\, I \in \mathcal{I} \setminus [\,m-1\,]^d\right) \approx \sum_{j=0}^{\nu-1} E\big(X_I \,|\, I \in \mathcal{I}_j\big)\ \mathsf{Prob}\left(I \in \mathcal{I}_j \,|\, I \in \mathcal{I} \setminus [\,m-1\,]^d\right) \ . \quad \text{(D.10)}$$

Since all the vectors in $\mathfrak{I}_j$ are assigned for all $j \in [\,\nu\,]$, we have $X_I = (m-2)j + (m-1)(d-j)$ for all $I \in \mathfrak{I}_j$. Substituting this result in (D.10) and simplifying the equations, we get

$$E\left(X_I \,|\, I \in \mathcal{I} \setminus [\,m-1\,]^d\right) \approx \frac{m(d-1) + \nu\theta^{\nu-1} + m(1-d)\theta^{\nu}}{n-(m-1)^d}\,(m-1)m^{d-1} \ . \tag{D.11}$$

Thus, we have

$$P_{lk} \approx \frac{\big(d(m-2)+\nu\big)(m-1)^d}{n(n-1)} + \frac{\theta m^d\big(m(d-1) + \nu\theta^{\nu-1} + m(1-d)\theta^{\nu}\big)}{n(n-1)} \ . \tag{D.12}$$

BIBLIOGRAPHY

[1] V. Acciaro, *On the complexity of computing Gröbner bases in characteristic* 2, Inform. Process. Lett., 51 (1994), pp. 321–323.

[2] L. M. Adleman and J. DeMarrais, *A subexponential algorithm for discrete logarithms over all finite fields*, in Proc. Adv. Cryptol. - CRYPTO'93, D. R. Stinson, ed., Berlin, 1993, Springer-Verlag, pp. 147–158.

[3] M.-L. Akkar, N. T. Courtois, R. Duteuil, and L. Goubin, *A fast and secure implementation of Sflash*, in Proc. Int. Workshop Practice and Theory in Public Key Crypto. - PKC'03, vol. 2567 of Lect. Notes Comput. Sci., Berlin, 2003, Springer-Verlag, pp. 267–278.

[4] M.-L. Akkar and J. Patarin, *Multivariate cryptography on smart cards.* Smart Card Security Conf.,Available at: `http://www.nmda.or.jp/nmda/ic-card/proceedings/Day2E.html`, Mar. 2001.

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, *A survey on sensor networks*, IEEE Commun. Mag., 40 (2002), pp. 102–114.

[6] R. Anderson and M. Kuhn, *Tamper resistance—a cautionary note*, in Proc. USENIX Workshop Elect. Commerce, Berkeley, CA, Nov. 1996, USENIX Assoc., pp. 1–11.

[7] ——, *Low cost attacks on tamper resistant devices*, in Proc. Int. Workshop Secur. Protocols, B. Christianson et al., eds., vol. 1361 of Lect. Notes Comput. Sci., Berlin, Apr. 1998, Springer-Verlag, pp. 125–136.

[8] ANSI X9.62, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, American National Standards Institute, NY, 1998.

[9] T. Arampatzis, J. Lygeros, and S. Manesis, *A survey of applications of wireless sensors and wireless sensor networks*, in Proc. IEEE Int. Symp. Intelligent Control, vol. 1, Limassol, Cyprus, June 2005, IEEE, pp. 719–724.

[10] F. Armknecht, M. Krause, and D. Stegemann, *Design principles for combiners with memory*, in Proc. Adv. Cryptol. - INDOCRYPT'05, S. Maitra et al., eds., vol. 3797 of Lect. Notes Comput. Sci., Berlin, 2005, Springer-Verlag, pp. 104–117.

[11] G. Ars, J.-C. Faugère, H. Imai, M. Kawazoe, and M. Sugita, *Comparison between XL and Gröbner basis algorithms*, in Proc. Adv. Cryptol. - ASIACRYPT'04, P. J. Lee, ed., vol. 3329 of Lect. Notes Comput. Sci., Berlin, 2004, Springer-Verlag, pp. 338–353.

[12] E. Ayday, F. Delgosha, and F. Fekri, *Location-aware security services for wireless sensor networks using network coding*, in Proc. IEEE Conf. Comput. Commun. - INFOCOM'07, AK, May 2007. CD-ROM.

[13] M. Bardet, *On the complexity of a Gröbner basis algorithm*, algorithms seminar, INRIA, 2002–2004. Available at: `http://algo.inria.fr/seminars/summary/Bardet2002.pdf`.

[14] T. Becker and V. Weispfenning, *Gröbner Bases: A Computational Approach to Commutative Algebra*, vol. 141 of Graduate Texts in Mathematics, Springer-Verlag, NY, 1993.

[15] M. Bellare and P. Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, in Proc. ACM Conf. Computer and Communications Security - CCS'93, ACM, NY, 1993, ACM Press, pp. 62–73.

[16] ——, *Introduction to modern cryptography.* Available at: `http://www-cse.ucsd.edu/~mihir/cse207/classnotes.html`, 2005.

[17] T. Beth and F. C. Piper, *The stop-and-go generator*, in Proc. Adv. Cryptol. - EUROCRYPT'84, T. Beth, N. Cot, and I. Ingemarsson, eds., vol. 209 of Lect. Notes Comput. Sci., Berlin, 1984, Springer-Verlag, pp. 88–92.

[18] E.-O. Blass and M. Zitterbart, *Towards acceptable public-key encryption in sensor networks*, in Proc. Int. Workshop on Ubiquitous Comput. - IWUC'05, S. K. Mostéfaoui and Z. Maamar, eds., INSTICC Press, 2005, pp. 88–93.

[19] R. Blom, *Non-public key distribution*, in Proc. Adv. Cryptol. - CRYPTO'82, D. Chaum, R. L. Rivest, , and A. T. Sherman, eds., NY, 1982, Plenum Publishing, pp. 231–236.

[20] ——, *An optimal class of symmetric key generation systems*, in Proc. Adv. Cryptol. - EUROCRYPT'84, T. Beth, N. Cot, and I. Ingemarsson, eds., vol. 209 of Lect. Notes Comput. Sci., Berlin, 1984, Springer-Verlag, pp. 335–338.

[21] M. Blum and S. Goldwasser, *An efficient probabilistic public key encryption scheme which hides all partial information*, in Proc. Adv. Cryptol. - CRYPTO'84, G. R. Blakley and D. Chaum, eds., vol. 196 of Lect. Notes Comput. Sci., Berlin, 1984, Springer-Verlag, pp. 289–302.

[22] C. Blundo et al., *Perfectly-secure key distribution for dynamic conferences*, in Proc. Adv. Cryptol. - CRYPTO'92, E. F. Brickell, ed., vol. 740 of Lect. Notes Comput. Sci., Berlin, 1992, Springer-Verlag, pp. 471–486.

[23] D. Boneh and G. Durfee, *Cryptanalysis of RSA with private key d less than* $N^{0.292}$, IEEE Trans. Inform. Theory, IT-46 (2000), pp. 1339–1349.

[24] D. Boneh, G. Durfee, and Y. Frankel, *An attack on RSA given a small fraction of the private key bits*, in Proc. Adv. Cryptol. - ASIACRYPT'98, K. Ohta and D. Pei, eds., vol. 1514 of Lect. Notes Comput. Sci., Berlin, 1998, Springer-Verlag, pp. 25–34.

[25] D. Boneh, B. Lynn, and H. Shacham, *Short signatures from the Weil pairing*, in Proc. Adv. Cryptol. - ASIACRYPT'01, C. Boyd, ed., vol. 2248 of Lect. Notes Comput. Sci., Berlin, 2001, Springer-Verlag, pp. 514–532.

[26] D. Boneh and R. Venkatesan, *Breaking RSA may not be equivalent to factoring*, in Proc. Adv. Cryptol. - EUROCRYPT'98, vol. 1233 of Lect. Notes Comput. Sci., Berlin, 1998, Springer-Verlag, pp. 59–71. Available at: `http://crypto.stanford.edu/~dabo/abstracts/no_rsa_red.html`.

[27] J. Borst, B. Preneel, and J. Vandwalle, *On the time-memory tradeoff between exhaustive key search and table precomputation*, in Symp. Inform. Theory in the Benelux, P. H. N. de With and M. van der Schaar-Mitrea, eds., Werkgemeenschap Informatie- en Communicatietheorie, Enschede (NL), 1998, pp. 111–118. Available at: `http://citeseer.ist.psu.edu/borst98timememory.html`.

[28] R. P. Brent, *Recent progress and prospects for integer factorization algorithms*, in Computing and Combinatorics - COCOON'00, D.-Z. Du et al., eds., vol. 1858 of Lect. Notes Comput. Sci., Springer-Verlag, Berlin, 2000, pp. 3–22.

[29] B. Büchberger, *A theoretical basis for the reduction of polynomials to canonical forms*, ACM SIGSAM Bull., 10 (1976), pp. 19–29.

[30] G. Caire, R. L. Grossman, and H. V. Poor, *Wavelet transforms associated with finite cyclic groups*, IEEE Trans. Inform. Theory, IT-39 (1993), pp. 1157–1166.

[31] E. Cardoza, R. Lipton, and A. R. Meyer, *Exponential space complete problems for petri nets and commutative semigroups (preliminary report)*, in Proc. Ann. ACM Symp. Theory of Computing - STOC'76, NY, 1976, ACM Press, pp. 50–54.

[32] H. Chan, A. Perrig, and D. Song, *Random key predistribution schemes for sensor networks*, in Proc. Symp. Security and Privacy, CA, 2003, IEEE Comput. Soc., pp. 197–213.

[33] K. S. Chan and F. Fekri, *A block cipher cryptosystem using wavelet transforms over finite fields*, IEEE Trans. Signal Process. Supp. Secure Media, 52 (2004), pp. 2975–2991.

[34] J.-M. Chen and B.-Y. Yang, *A more secure and efficacious TTS signature scheme*. Cryptol. ePrint Archive, Report 2003/160, 2003. Available at: `http://eprint.iacr.org/2003/160`.

[35] S. Cheng and Z. Xiong, *Audio coding and image denoising based on the nonuniform modulated complex lapped transform*, IEEE Trans. Multimedia, 7 (2005), pp. 817–827.

[36] D.-M. Chung and Y. Wang, *Lapped orthogonal transforms designed for error-resilient image coding*, IEEE Trans. Circuits Syst. Video Technol., 12 (2002), pp. 752–764.

[37] D. Coppersmith, H. Krawczyk, and Y. Mansour, *The shrinking generator*, in Proc. Adv. Cryptol. - CRYPTO'93, D. R. Stinson, ed., vol. 773 of Lect. Notes Comput. Sci., Berlin, 1993, Springer-Verlag, pp. 22–39.

[38] D. Coppersmith, J. Stern, and S. Vaudenay, *Attacks on the birational permutation signature schemes*, in Proc. Adv. Cryptol. - CRYPTO'93, D. R. Stinson, ed., vol. 773 of Lect. Notes Comput. Sci., Berlin, 1994, Springer-Verlag, pp. 435–443.

[39] D. Coppersmith, J. Stern, and S. Vaudenay, *The security of the birational permutation signature schemes*, J. Cryptology, 10 (1997), pp. 207–221.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, The MIT Press, MA, 2nd ed., 2001.

[41] N. T. Courtois, *The security of hidden field equations (HFE)*, in Proc. Cryptographers' Track at the RSA Conf. - CT-RSA'01, D. Naccache, ed., vol. 2020 of Lect. Notes Comput. Sci., Berlin, 2001, Springer-Verlag, pp. 266–281.

[42] ——, *Generic attacks and the security of Quartz*, in Proc. Int. Workshop Practice and Theory in Public Key Crypto. - PKC'03, Y. G. Desmedt, ed., vol. 2567 of Lect. Notes Comput. Sci., Berlin, 2003, Springer-Verlag, pp. 351–364.

[43] N. T. Courtois, M. Daum, and P. Felke, *On the security of HFE, HFEv– and Quartz*, in Proc. Int. Workshop Practice and Theory in Public Key Crypto. - PKC'03, Y. G. Desmedt, ed., vol. 2567 of Lect. Notes Comput. Sci., Berlin, 2003, Springer-Verlag, pp. 337–350.

[44] N. T. Courtois, A. Klimov, J. Patarin, and A. Shamir, *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, in Proc. Adv. Cryptol. - EUROCRYPT'00, B. Preneel, ed., vol. 1807 of Lect. Notes Comput. Sci., Berlin, 2000, Springer-Verlag, pp. 392–407.

[45] D. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Undergraduate Texts in Mathematics, Springer-Verlag, NY, 2nd ed., 1997.

[46] ——, *Using Algebraic Geometry*, Graduate Texts in Mathematics, Springer-Verlag, NY, 1998.

[47] R. Crandall and C. Pomerance, *Prime Numbers: A Computational Perspective*, Springer-Verlag, NY, 2001.

[48] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*, Springer-Verlag, Berlin, 2002.

[49] I. Daubechies, *Orthonormal bases of compactly supported wavelets*, Commun. Pure and Appl. Math., 41 (1988), pp. 909–996.

[50] I. Daubechies, *The wavelet transform, time-frequency localization and signal analysis*, IEEE Trans. Inform. Theory, it-36 (1990), pp. 961–1005.

[51] I. Daubechies, *Ten lectures on wavelets*, SIAM, (1992).

[52] F. Delgosha, E. Ayday, K. Chan, and F. Fekri, *Security services in wireless sensor networks using sparse random coding*, in Proc. IEEE Commun. Soc. Conf. Sensor and Ad Hoc Commun. and Net. - SECON'06, VI, Apr. 2006. CD-ROM.

[53] F. Delgosha, E. Ayday, and F. Fekri, *MKPS: A multivariate polynomial scheme for symmetric key-establishment in distributed sensor networks*, in accepted in Int. Wirel. Commun. Mobile Comput. Conf. - IWCMC'07, ACM, Aug. 2007.

[54] F. Delgosha, K. Chan, and F. Fekri, *Symmetric cryptography using finite-field wavelets*, submitted to IEEE Trans. Circuits Syst. I, Reg. Papers, (2004).

[55] F. Delgosha, K. S. Chan, and F. Fekri, *Multivariate symmetric cryptography using finite-field wavelets*, in Proc. Hawaii and SITA Joint Conf. Inform. Theory - HISC'07, HI, May 2007, pp. 130–135.

[56] F. Delgosha and F. Fekri, *Factorization of two-channel 2D paraunitary filter banks over fields of characteristic two*, in Proc. IEEE Int. Symp. Inform. Theory - ISIT'04, IL, June-July 2004, p. 565.

[57] ——, *On the factorization of two-dimensional paraunitary filter banks*, in Proc. Int. Conf. Acoust. Speech Signal Process. - ICASSP'04, vol. 2, Montreal, Canada, May 2004, IEEE, pp. 969–972.

[58] F. Delgosha and F. Fekri, *Results on the factorization of multidimensional matrices for paraunitary filter banks over the complex field*, IEEE Trans. Signal Processing, 52 (2004), pp. 1289–1303.

[59] F. Delgosha and F. Fekri, *Key pre-distribution in wireless sensor networks using multivariate polynomials*, in Proc. IEEE Commun. Soc. Conf. Sensor and Ad Hoc Commun. and Net. - SECON'05, NJ, 2005, IEEE. CD-ROM.

[60] ——, *Stream cipher using finite-field wavelets*, in Proc. Int. Conf. Acoust. Speech Signal Process. - ICASSP'05, vol. 5, Philadelphia, PA, Mar. 2005, pp. 689–692.

[61] ——, *A multivariate key-establishment scheme for wireless sensor networks*, submitted to IEEE Trans. Wireless Commun., (2006).

[62] ——, *Multivariate signature using algebraic techniques*, in Proc. IEEE Int. Symp. Inform. Theory - ISIT'06, 2006. CD-ROM.

[63] ——, *Public-key cryptography using paraunitary matrices*, IEEE Trans. Signal Processing, 54 (2006), pp. 3489–3504.

[64] ——, *Threshold key-establishment in distributed sensor networks using a multivariate scheme*, in Proc. IEEE Conf. Comput. Commun. - INFOCOM'06, Barcelona, 2006, IEEE. CD-ROM.

[65] F. Delgosha, M. Sartipi, and F. Fekri, *Construction of two-dimensional paraunitary filter banks over fields of characteristic two and their connections to error-control coding*, submitted to IEEE Trans. Circuits Syst. I, Reg. Papers, (2006).

[66] M. Dickerson, *Polynomial decomposition algorithms for multivariate polynomials*, Technical Report TR87-826, Comput. Sci., Cornell Univ., NY, Apr. 1987.

[67] W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory, IT-22 (1976), pp. 644–654.

[68] C. Ding, G. Xiao, and W. Shan, *The Stability Theory of Stream Ciphers*, vol. 561 of Lect. Notes Comput. Sci., Springer-Verlag, Berlin, 1991.

[69] H. Dobbertin, *One-to-one highly nonlinear power functions on* $\mathsf{GF}(2^n)$, Appl. Algebra Eng. Commun. Comput., 9 (1998), pp. 139–152.

[70] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, *A key management scheme for wireless sensor networks using deployment knowledge*, in Proc. IEEE Conf. Comput. Commun. - INFOCOM'04, vol. 1, NJ, 2004, IEEE, pp. 586–597.

[71] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, *A pairwise key pre-distribution scheme for wireless sensor networks*, in Proc. ACM Conf. Computer Commun. Secur. - CCS'03, NY, 2003, ACM Press, pp. 42–51.

[72] W. Du, R. Wang, and P. Ning, *An efficient scheme for authenticating public keys in sensor networks*, in Proc. ACM Int. Symp. Mobile Ad Hoc Net. Comput. - MobiHoc'05, NY, 2005, ACM Press, pp. 58–67.

[73] D. Eisenbud, *Commutative Algebra with a View Toward Algebraic Geometry*, vol. 150 of Graduate Texts in Mathematics, Springer-Verlag, NY, 1995.

[74] T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. Inform. Theory, IT-31 (1985), pp. 469–472.

[75] H. Englund and T. Johansson, *A new simple technique to attack filter generators and related ciphers*, in Proc. Select. Areas Crypto. - SAC'04, H. Handschuh and A. Hasan, eds., vol. 3357 of Lect. Notes Comput. Sci., Berlin, 2005, Springer-Verlag, pp. 39–53.

[76] P. Erdős and A. Rényi, *On the evolution of random graphs*, Bull. Inst. Internat. Statist., 38 (1961), pp. 343–347.

[77] L. Eschenauer and V. D. Gligor, *A key-management scheme for distributed sensor networks*, in Proc. ACM Conf. Computer Commun. Secur. - CCS'02, DC, 2002, ACM Press, pp. 41–47.

[78] J.-C. Faugére, *A new efficient algorithm for computing Gröbner bases (F_4)*, J. Pure and Applied Algebra, 139 (1999), pp. 61–88.

[79] ——, *A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5)*, in Proc. Int. Symp. Symbolic and Algebraic Comput. - ISSAC'02, T. Mora, ed., NY, 2002, ACM, pp. 75–83.

[80] J.-C. Faugère and A. Joux, *Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases*, in Proc. Adv. Cryptol. - CRYPTO'03, D. Boneh, ed., vol. 2729 of Lect. Notes Comput. Sci., Berlin, 2003, Springer-Verlag, pp. 44–60.

[81] Federal Inform. Process. Standards, *DES Modes of Operation*, Dec. 1980. FIPS Pubs. 81.

[82] S. V. Fedorenko and P. V. Trifonov, *Finding roots of polynomials over finite fields*, IEEE Trans. Communn., 50 (2002), pp. 1709–1711.

[83] F. Fekri and F. Delgosha, *Finite-Field Wavelets and Their Applications in Cryptography and Coding*, Prentice-Hall, 2007.

[84] F. Fekri, R. Mersereau, and R. Schafer, *Theory of wavelet transform over finite fields*, Proc. Int. Conf. Acoust. Speech Signal Process. - ICASSP'99, 3 (1999), pp. 1213–1216.

[85] F. Fekri, R. M. Mersereau, and R. W. Schafer, *Theory of paraunitary filter banks over fields of characteristic two*, IEEE Trans. Inform. Theory, it-48 (2002), pp. 2964–2979.

[86] ——, *Two-band wavelets and filter banks over finite fields with connections to error control coding*, IEEE Trans. Signal Processing, 51 (2003), pp. 3143–3151.

[87] H. Fell and W. Diffie, *Analysis of a public key approach based on polynomial substitution*, in Proc. Adv. Cryptol. - CRYPTO'85, H. C. Williams, ed., vol. 218 of Lect. Notes Comput. Sci., Springer-Verlag, 1986, pp. 340–349.

[88] T. R. Gardos, *Analysis and Design of Multidimensional (FIR) Filter Banks*, ph.d. dissertation, Dept. Elect. Eng., Georgia Inst. Technol., June 1993.

[89] J. V. Z. Gathen, *Functional decomposition of polynomials: the tame case*, J. Symb. Comput., 9 (1990), pp. 281–299.

[90] ——, *Functional decomposition of polynomials: the wild case*, J. Symb. Comput., 10 (1990), pp. 437–452.

[91] G. Gaubatz, E. O. Jens-Peter Kaps, and B. Sunar, *State of the art in ultra-low power public key cryptography for wireless sensor networks*, in Proc. IEEE Int. Conf. Pervasive Comput. Commun. - PerCom'05, HI, Mar. 2005, IEEE Comput. Soc., pp. 146–150.

[92] G. Gaubatz, J.-P. Kaps, and B. Sunar, *Public key cryptography in sensor networks–revisited*, in Proc. Secur. in Ad-hoc and Sensor Net. - ESAS'04, C. Castelluccia et al., eds., vol. 3313 of Lect. Notes Comput. Sci., Berlin, 2005, Springer-Verlag, pp. 2–18.

[93] H. Gilbert and M. Minier, *Cryptanalysis of SFLASH*, in Proc. Adv. Cryptol. - EUROCRYPT'02, L. R. Knudsen, ed., vol. 2332 of Lect. Notes Comput. Sci., Berlin, 2002, Springer-Verlag, pp. 288–298.

[94] L. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, Academic Press, NY, 1982.

[95] S. Goldwasser and M. Bellare, *Lecture notes on cryptography.* Available at: `http://www.cs.ucsd.edu/users/mihir/papers/gb.html`, 2001.

[96] S. Goldwasser and S. Micali, *Probabilistic encryption*, in J. Comput. Syst. Sci., vol. 28, CA, 1984, ACM Press, pp. 270–299.

[97] D. Gollmann and W. Chambers, *Clock-controlled shift registers: A review*, IEEE J. Select. Areas Commun., 7 (1989), pp. 525–533.

[98] L. Goubin and N. T. Courtois, *Cryptanalysis of the TTM cryptosystem*, in Proc. Adv. Cryptol. - ASIACRYPT'00, T. Okamoto, ed., vol. 1976 of Lect. Notes Comput. Sci., Berlin, 2000, Springer-Verlag, pp. 44–57.

[99] G.-M. Greuel, G. Pfister, and H. Schönemann, Singular *3.0*, A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern, 2005. Available at: `http://www.singular.uni-kl.de`.

[100] L. C. Guillou and M. Ugon, *Smart card, a highly reliable and portable security device*, in Proc. Adv. Cryptol. - CRYPTO'86, A. M. Odlyzko, ed., vol. 263 of Lect. Notes Comput. Sci., Berlin, 1986, Springer-Verlag, pp. 464–479.

[101] C. G. Günther, *Alternating step generators controlled by de Bruijn sequences*, in Proc. Adv. Cryptol. - EUROCRYPT'87, D. Chaum and W. L. Price, eds., vol. 304 of Lect. Notes Comput. Sci., Berlin, 1987, Springer-Verlag, pp. 5–14.

[102] M. E. Hellman, *A cryptanalytic time-memory trade-off*, IEEE Trans. Inform. Theory, IT-26 (1980), pp. 401–406.

[103] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Computer Science, Addison Wesley, 1979.

[104] Y. Hu and G. Xiao, *Generalized self-shrinking generator*, IEEE Trans. Inform. Theory, 50 (2004), pp. 714–719.

[105] T. W. Hungerford, *Algebra*, vol. 73 of Graduate Texts in Mathematics, Springer-Verlag, NY, 1974.

[106] J. Hwang and Y. Kim, *Revisiting random key pre-distribution schemes for wireless sensor networks*, in Proc. ACM Workshop Secur. Ad Hoc Sens. Net. - SASN'04, NY, 2004, ACM Press, pp. 43–52.

[107] T. Jakobsen, *Attacks on block ciphers of low algebraic degree*, J. Cryptology, 14 (2001), pp. 197–210.

[108] T. Jakobsen and L. R. Knudsen, *The interpolation attack on block ciphers*, in Proc. Fast Softw. Encrypt. - FSE'97, E. Biham, ed., vol. 1267 of Lect. Notes Comput. Sci., Berlin, 1997, Springer-Verlag, pp. 28–40.

[109] S. Janson, T. Łuczak, and A. Rucińki, *Random Graphs*, Disc. Math. Optim., John Wiley, NY, 2000.

[110] S. T. Jivkova and M. Kavehrad, *Transceiver design concept for cellular and multispot diffusing regimes of transmission*, EURASIP J. Appl. Signal Processing, (2005), pp. 30–38.

[111] T. Kaller, H. Park, and M. Vetterli, *Gröbner basis techniques in multidimensional multirate systems*, Proc. Int. Conf. Acoust. Speech Signal Process. - ICASSP'95, 4 (1995), pp. 2121–2124.

[112] C. Karlof and D. Wagner, *Secure routing in wireless sensor networks: attacks and countermeasures*, in Proc. Int. Workshop Sens. Net. Protocols and App. - SNPA'03, NJ, May 2003, IEEE, pp. 113–127.

[113] G. Karlsson and M. Vetterli, *Theory of two-dimensional multirate filter banks*, IEEE Trans. Acoust., Speech, Signal Process., 38 (1990), pp. 925–937.

[114] S. H. Kim and C. S. Leem, *Security threats and their countermeasures of mobile portable computing devices in ubiquitous computing environments*, in Proc. Int. Conf. Comput. Sci. and Its Appl. - ICCSA'05, O. Gervasi et al., eds., vol. 3483 of Lect. Notes Comput. Sci., Berlin, 2005, Springer-Verlag, pp. 79–85.

[115] A. Kipnis, J. Patarin, and L. Goubin, *Unbalanced oil and vinegar signature schemes*, in Proc. Adv. Cryptol. - EUROCRYPT'99, J. Stern, ed., vol. 1592 of Lect. Notes Comput. Sci., Berlin, 1999, Springer-Verlag, pp. 206–222.

[116] A. Kipnis and A. Shamir, *Cryptanalysis of the HFE public key cryptosystem by relinearization*, in Proc. Adv. Cryptol. - CRYPTO'99, M. Wiener, ed., vol. 1666 of Lect. Notes Comput. Sci., Berlin, 1999, Springer-Verlag, pp. 19–30.

[117] N. Koblitz, *Elliptic curve cryptosystems*, Math. Comp., 48 (1987), pp. 203–209.

[118] N. Koblitz, *Algebraic Aspects of Cryptography*, vol. 3 of Algorithms and Computation in Mathematics, Springer-Verlag, Berlin, 1998.

[119] N. Koblitz and A. J. Menezes, *A survey of public-key cryptosystems*, SIAM Rev., 46 (2004), pp. 599–634.

[120] D. Kozen, S. Landau, and R. Zippel, *Decomposition of algebraic functions*, J. Symb. Comput., 22 (1996), pp. 235–246.

[121] K. Kühnle and E. W. Mayr, *Exponential space computation of Gröbner bases*, in Proc. Int. Symp. Symb. Algebraic Comput. - ISSAC'96, NY, 1996, ACM Press, pp. 63–71.

[122] S. Lange, *Algebra*, Addison-Wesley, NY, 3rd ed., 1993.

[123] L. Lazos, R. Poovendran, and S. Čapkun, *ROPE: Robust position estimation in wireless sensor networks*, in Proc. Int. Symp. Inform. Process. Sensor Net. - IPSN'05, CA, 2005, IEEE, pp. 324–331.

[124] S.-Y. R. Li et al., *Linear network coding*, IEEE Trans. Inform. Theory, 49 (2003), pp. 371–381.

[125] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and their Applications*, Cambridge University Press, UK, 1994.

[126] D. Liu and P. Ning, *Location-based pairwise key establishments for static sensor networks*, in Proc. ACM Workshop Secur. Ad Hoc Sens. Net. - SASN'03, VA, 2003, ACM Press, pp. 72–82.

[127] D. Liu, P. Ning, and R. Li, *Establishing pairwise keys in distributed sensor networks*, ACM Trans. Inf. Syst. Secur., 8 (2005), pp. 41–77.

[128] V. C. Liu and P. P. Vaidyanathan, *On factorization of a subclass of 2-D digital FIR lossless matrices for 2-D QMF bank applications*, IEEE Trans. Circuits Syst., 37 (1990), pp. 852–854.

[129] Y. Liu, B. Ni, X. Feng, and E. J. Delp, *Lapped-orthogonal-transform-based adaptive image watermarking*, J. Electron. Imaging, 15 (2006), p. 013009.

[130] I. Mantin, *Analysis of the stream cipher RC4*, master's thesis, The Weizmann Ins. of Sience, Israel, Nov. 2001. Available at: `http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html`.

[131] G. Martinet, *Quartz, Flash and SFlash*, NESSIE Public Rep., (2001). Available at: `https://www.cosic.esat.kuleuven.ac.be/nessie/reports/`.

[132] T. Matsumoto and H. Imai, *Public quadratic polynomial-tuples for efficient signature-verification and message-encryption*, in Proc. Adv. Cryptol. - EUROCRYPT'88, C. G. Guenther, ed., vol. 330 of Lect. Notes Comput. Sci., Springer-Verlag, 1988, pp. 419–453.

[133] S. Maubach, *Polynomial automorphisms over finite fields*, Serdica Math. J., 27 (2001), pp. 343–350.

[134] U. M. Maurer, *New approaches to the design of self-synchronizing stream ciphers*, in Proc. Adv. Cryptol. - EUROCRYPT'91, D. W. Davies, ed., vol. 547 of Lect. Notes Comput. Sci., Berlin, 1991, Springer-Verlag, pp. 458–471.

[135] W. Meier and O. Staffelbach, *The self-shrinking generator*, in Proc. Adv. Cryptol. - EUROCRYPT'95, A. D. Santis, ed., vol. 950 of Lect. Notes Comput. Sci., Berlin, 1995, Springer-Verlag, pp. 205–214.

[136] A. J. Menezes et al., *Applications of Finite Fields*, Kluwer Academic, MA, 1993.

[137] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, NY, 1997.

[138] R. C. Merkle, *A digital signature based on a conventional encryption function*, in Proc. Adv. Cryptol. - CRYPTO'87, C. Pomerance, ed., vol. 293 of Lect. Notes Comput. Sci., Berlin, 1987, Springer-Verlag, pp. 369–378.

[139] M. Mihaljević and J. D. Golić, *A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence*, in Proc. Adv. Cryptol. - AUSCRYPT'90, J. Seberry and J. Pieprzyk, eds., vol. 453 of Lect. Notes Comput. Sci., Berlin, 1990, Springer-Verlag, pp. 165–175.

[140] A. Miller, *PDA security concerns*, Netw. Secur., (2004), pp. 8–10.

[141] V. Miller, *Use of elliptic curves in cryptography*, in Proc. Adv. Cryptol. - CRYPTO'85, H. C. Williams, ed., vol. 218 of Lect. Notes Comput. Sci., Springer-Verlag, 1986, pp. 417–426.

[142] N. Modadugu, D. Boneh, and M. Kim, *Generating RSA keys on a handheld using an untrusted server*, in Proc. Adv. Cryptol. - INDOCRYPT'00, B. K. Roy and E. Okamoto, eds., vol. 1977 of Lect. Notes Comput. Sci., Berlin, 2000, Springer-Verlag, pp. 271–282.

[143] T. T. Moh, *A fast public key system with signature and master key functions*, in Proc. Int. Workshop on Crypto. Tech. E-Commerce - CrypTEC'99, Kowloon, Hong Kong, 1999, City Univ. Hong Kong, pp. 63–69. Available at: `http://www.usdsi.com/cryptec.ps`.

[144] ——, *A public key system with signature and master key functions*, Commununications in Algebra, 27 (1999), pp. 2207–2222. Available at: `http://www.usdsi.com/public.ps`.

[145] M. Morf, B. C. Levy, and S. Y. Kung, *New results in 2-D systems theory, Part I: 2-D polynomial matrices, factorization, and coprimeness*, Proc. IEEE, 65 (1977), pp. 861–872.

[146] S. Moriai, T. Shimoyama, and T. Kaneko, *An efficient interpolation attack*, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. (Japan), E83-A (2000), pp. 39–47.

[147] M. Noorkami and F. Fekri, *A fast correlation attack via unequal error correcting LDPC codes*, in Proc. Cryptographers' Track at the RSA Conf. - CT-RSA'04, T. Okamoto, ed., vol. 2964 of Lect. Notes Comput. Sci., Berlin, 2004, Springer-Verlag, pp. 54–66.

[148] K. Nyberg, *Differentially uniform mappings for cryptography*, in Proc. Adv. Cryptol. - EUROCRYPT'93, T. Helleseth, ed., vol. 765 of Lect. Notes Comput. Sci., Berlin, 1993, Springer-Verlag, pp. 55–64.

[149] ——, *S-boxes and round functions with controllable linearity and differential uniformity*, in Proc. Fast Softw. Encrypt. - FSE'95, B. Preneel, ed., vol. 1008 of Lect. Notes Comput. Sci., Berlin, 1995, Springer-Verlag, pp. 111–129.

[150] P. Oechslin, *Making a faster cryptanalytic time-memory trade-off*, in Proc. Adv. Cryptol. - CRYPTO'03, D. Boneh, ed., vol. 2729 of Lect. Notes Comput. Sci., Berlin, 2003, Springer-Verlag, pp. 617–630.

[151] H. Park, *2-D non-separable paraunitary matrices and Gröbner bases*, Proc. IEEE Int. Symp. Circuits Syst. - ISCS'01, 2 (2001), pp. 441–444.

[152] H.-J. Park, *A Computational Theory of Laurent Polynomial Rings and Multidimensional FIR Systems*, Ph.D. dissertation, College Eng., Univ. Calif., Berkeley, May 1995.

[153] J. Patarin, *Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88*, in Proc. Adv. Cryptol. - CRYPTO'95, D. Coppersmith, ed., vol. 963 of Lect. Notes Comput. Sci., Berlin, 1995, Springer-Verlag, pp. 248–261.

[154] ——, *Asymmetric cryptography with a hidden monomial*, in Proc. Adv. Cryptol. - CRYPTO'96, N. Koblitz, ed., vol. 1109 of Lect. Notes Comput. Sci., Berlin, 1996, Springer-Verlag, pp. 45–60.

[155] ——, *Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms*, in Proc. Adv. Cryptol. - EUROCRYPT'96, U. Maurer, ed., vol. 1070 of Lect. Notes Comput. Sci., Berlin, 1996, Springer-Verlag, pp. 33–48.

[156] J. Patarin, N. T. Courtois, , and L. Goubin, *FLASH, a fast multivariate signature algorithm,* `http://www.minrank.org/flash/`, in Proc. Cryptographers' Track at the RSA Conf. - CT-RSA'01, D. Naccache, ed., vol. 2020 of Lect. Notes Comput. Sci., Berlin, 1999, Springer-Verlag, pp. 298–307.

[157] ——, *QUARTZ, 128-bit long digital signatures,* `http://www.minrank.org/quartz/`, in Proc. Cryptographers' Track at the RSA Conf. - CT-RSA'01, D. Naccache, ed., vol. 2020 of Lect. Notes Comput. Sci., Berlin, 2001, Springer-Verlag, pp. 282–297.

[158] J. Patarin, L. Goubin, and N. T. Courtois, *C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*, in Proc. Adv. Cryptol. - ASIACRYPT'98, K. Ohta and D. Pei, eds., vol. 1514 of Lect. Notes Comput. Sci., Berlin, 1998, Springer-Verlag, pp. 35–49.

[159] A. Perrig and J. D. Tygar, *Secure Broadcast Communication in Wired and Wireless Networks*, Kluwer Academic, Boston, 2003.

[160] S. M. Phoong and P. P. Vaidyanathan, *New results on paraunitary filter banks over finite fields*, IEEE Int. Symp. Circuits Syst., 2 (1996), pp. 413–416.

[161] ——, *Paraunitary filter banks over finite fields*, IEEE Trans. Signal Processing, 45 (1997), pp. 1443–1457.

[162] H. Pishro-Nik, K. Chan, and F. Fekri, *On connectivity properties of large-scale wireless sensor networks*, in Proc. First Ann. IEEE Commun. Society Conf. on Sensor Commun. and Net., Santa Clara, CA, 4-7 October 2004. CD-ROM.

[163] B. Preneel, *A survey of recent developments in cryptographic algorithms for smart cards*, Comput. Networks, 51 (2007), pp. 2223–2233.

[164] B. Preneel et al., *NESSIE security report*, Report NES/DOC/ENS/WP5/D20/2, NESSIE, Feb. 2003. Available at: `https://www.cosic.esat.kuleuven.be/nessie/deliverables/D20-v2.pdf`.

[165] ——, *Performance of optimized implementations of the NESSIE primitives*, Report NES/DOC/TEC/WP6/D21/2, NESSIE, Feb. 2003. Available at: `https://www.cosic.esat.kuleuven.be/nessie/deliverables/D21-v2.pdf`.

[166] K. Ren, W. Lou, and Y. Zhang, *LEDS: Providing location-aware end-to-end data security in wireless sensor networks*, in Proc. IEEE Conf. Comput. Commun. - INFOCOM'06, 2006.

[167] R. L. Rivest, A. Shamir, and L. M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM (USA), 21 (1978), pp. 120–126.

[168] M. Robshaw, *Stream ciphers*, Tech. Rep. TR-701, RSA Security Inc., CA, July 1995. Available at: `ftp://ftp.rsasecurity.com/pub/pdfs/tr701.pdf`.

[169] K. H. Rosen, *Elementary Number Theory and its Applications*, Addison Wesley Longman, 4th ed., 2000.

[170] R. A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer-Verlag, NY, 1986.

[171] M. Sartipi, F. Delgosha, and F. Fekri, *Two-dimensional half-rate codes using two-variable finite-field filter banks*, IEEE Trans. Signal Processing, (2006).

[172] M. Sartipi and F. Fekri, *Two-dimensional error correcting codes using finite-field wavelets*, in Proc. Inform. Theory Workshop, San Antonio, TX, Oct. 2004, pp. 22–27.

[173] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, 2nd ed., 1996.

[174] I. R. Shafarevich, *Basic Algebraic Geometry 1: Varieties in Projective Space*, Springer-Verlag, Berlin, 2nd ed., 1994.

[175] A. Shamir, *Efficient signature schemes based on birational permutations*, in Proc. Adv. Cryptol. - CRYPTO'93, D. R. Stinson, ed., vol. 773 of Lect. Notes Comput. Sci., Berlin, 1994, Springer-Verlag, pp. 1–12.

[176] C. E. Shannon, *Communication theory of secrecy systems*, Bell Syst. Tech. J., 28 (1949), pp. 656–715.

[177] T. Siegenthaler, *Correlation-immunity of non-linear combining functions for cryptographic applications*, IEEE Trans. Inform. Theory, it-30 (1984), pp. 776–780.

[178] L. R. Simpson et al., *LILI keystream generator*, in Proc. Select. Areas Crypto. - SAC'00, D. Stinson and S. Tavares, eds., vol. 2012 of Lect. Notes Comput. Sci., Berlin, 2000, Springer-Verlag, pp. 248–261.

[179] D. R. Stinson, *Cryptography: Theory and Practice*, The CRC Series on Discrete Mathematics and its Applications, CRC Press, 1995.

[180] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, MA, 1996.

[181] G. L. Stüber, *Principles of Mobile Communication*, Kluwer Academic, Boston, 2nd ed., 2001.

[182] M. Szydlo, *Merkle tree traversal in log space and time*, in Proc. Adv. Cryptol. - EUROCRYPT'04, C. Cachin and J. Camenisch, eds., vol. 3027 of Lect. Notes Comput. Sci., Berlin, May 2004, Springer-Verlag, pp. 541–554.

[183] P. P. Vaidyanathan, *Theory and design of M-channel maximally decimated quadrature mirror filters with arbitrary M, having the perfect-reconstruction property*, IEEE Trans. Acoust., Speech, Signal Process., ASSP-35 (1987), pp. 476–492.

[184] ——, *A tutorial on multirate digital filter banks*, Proc. IEEE Int. Symp. Circuits Syst. - ISCS'88, 3 (1988), pp. 2241–2248.

[185] ——, *Unitary and paraunitary systems in finite fields*, IEEE. Int. Symp. Circuits Syst., 2 (1990), pp. 1189–1192.

[186] ——, *Multirate Systems and Filter Banks*, Prentice-Hall, NJ, 1993.

[187] A. van den Essen, *Polynomial Automorphisms and the Jacobian Conjecture*, vol. 190 of Progress in Mathematics, Birkhäuser Verlag, Berlin, 2000.

[188] S. Venkataraman and B. Levy, *A comparison of design methods for 2-D FIR orthogonal perfect reconstruction filter banks*, IEEE Trans. Circuits Syst.II, 42 (1995), pp. 525–536.

[189] M. Vetterli, *Multi-dimensional sub-band coding: Some theory and algorithms*, Signal Process., 6 (1984), pp. 97–112.

[190] M. Vetterli and J. Kovačević, *Wavelets and Subband Coding*, Prentice-Hall, NJ, 1995.

[191] E. Viscito and J. Allebach, *Design of perfect reconstruction multidimensional filter banks using cascaded smith form matrices*, Proc. IEEE Int. Symp. Circuits Syst. - ISCS'88, 1 (1988), pp. 831–834.

[192] J. von zur Gathen, J. Gutierrez, and R. Rubio, *Multivariate polynomial decomposition*, Appl. Algebra Eng. Commun. Comput., 14 (2003), pp. 11–31.

[193] D. Wagner, *Resilient aggregation in sensor networks*, in Proc. ACM Workshop Secur. Ad Hoc Sens. Net. - SASN'04, NY, 2004, ACM Press, pp. 78–87.

[194] A. S. Wander et al., *Energy analysis of public-key cryptography for wireless sensor networks*, in Proc. IEEE Int. Conf. Pervasive Comput. Commun. - PerCom'05, CA, Mar. 2005, IEEE Comput. Soc., pp. 324–328.

[195] L.-C. Wang and F.-H. Chang, *Revision of tractable rational map cryptosystem*. Cryptol. ePrint Archive, Report 2004/046, 2004. Available at: `http://eprint.iacr.org/2004/046`.

[196] I. Watson, *Securing portable storage devices*, Netw. Secur., (2006), pp. 8–11.

[197] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall, NJ, 1994.

[198] Wikipedia, *Reciprocal polynomial — wikipedia, the free encyclopedia*. Available at: `http://en.wikipedia.org/w/index.php?title=Reciprocal_polynomial&oldid=97003759` (May 2007).

[199] ——, *Sesquilinear form — wikipedia, the free encyclopedia*. Available at: `http://en.wikipedia.org/w/index.php?title=Sesquilinear_form&oldid=112720259` (May 2007).

[200] E. D. Win, S. Mister, B. Preneel, and M. Wiener, *On the performance of signature schemes based on elliptic curves*, in Proc. Algorithmic Number Theory Symp. III - ANTS-III, J. P. Buhler, ed., vol. 1423 of Lect. Notes Comput. Sci., Berlin, 1998, Springer-Verlag, pp. 252–266.

[201] C. Wolf, P. Fitzpatrick, S. N. Foley, and E. Popovici, *HFE in Java: implementing hidden field for public key cryptography*, in Proc. Irish Signals and Syst. Conf. - ISSC'02, Cork, Ireland, June 2002, Cork Inst. Technol, pp. 295–299.

[202] S. Wolfram, *Cryptography with cellular automata*, in Proc. Adv. Cryptol. - CRYPTO'85, H. C. Williams, ed., vol. 218 of Lect. Notes Comput. Sci., Berlin, 1986, Springer-Verlag, pp. 429–432.

[203] B.-Y. Yang and J.-M. Chen, *All in the XL family: theory and practice*, in Proc. Int. Conf. Inform. Security and Cryptology - ICISC'04, C. Park and S. Chee, eds., vol. 3506 of Lect. Notes Comput. Sci., Berlin, 2004, Springer-Verlag, pp. 67–86.

[204] ——, *TTS: Rank attacks in tame-like multivariate PKCs*. Cryptol. ePrint Archive, Report 2004/061, 2004. Available at: `http://eprint.iacr.org/2004/061`.

[205] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, *Toward resilient security in wireless sensor networks*, in Proc. ACM Int. Symp. Mobile Ad Hoc Net. Comput. - MobiHoc'05, NY, 2005, ACM Press, pp. 34–45.

[206] F. Ye, H. Luo, S. Lu, and L. Zhang, *Statistical en-route filtering of injected false data in sensor networks*, IEEE J. Sel. Areas Commun., 23 (2005), pp. 839–850.

[207] Y. Zhang, R. W. Yeung, and N. Cai, *Location-based compromise-tolerant security mechanisms for wireless sensor networks*, IEEE J. Select. Areas Commun., 24 (2006), pp. 247–260.

[208] S. Zhu, S. Setia, S. Jajodia, and P. Ning, *An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks*, in Proc. IEEE Symp. Secur. and Privacy, CA, May 2004, IEEE Comput. Soc., pp. 259–271.

VITA

Farshid Delgosha received the B.Sc. degree in 1995 from Tehran Polytechnic and M.Sc. degree in 1997 from Sharif University of Technology all in Tehran, Iran. In fall 2001, he started his Ph.D. studies at the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA. He will join New York Institute of Technology as an Assistant Professor starting Fall 2007.

His research interests lie in the general areas of cryptography, network security (wireless sensor and ad-hoc networks), signal processing, and communications. For his research at Georgia Tech, he received the Outstanding Research Award from the Center of Signal and Image Processing in spring 2006.