QUAD: Overview and Recent Developments

David Arditti[1], Côme Berbain[1], Olivier Billet[1],
Henri Gilbert[1], and Jacques Patarin[2]

[1] France Telecom Research and Development,
38-40 rue du Général Leclerc, F-92794 Issy-les-Moulineaux, France.
`firstname.lastname@orange-ftgroup.com`
[2] Université de Versailles,
45 avenue des Etats-Unis, F-78035 Versailles cedex, France.
`jacques.patarin@prism.uvsq.fr`

Abstract. We give an outline of the specification and provable security features of the QUAD stream cipher proposed at Eurocrypt 2006 [5]. The cipher relies on the iteration of a multivariate system of quadratic equations over a finite field, typically GF(2) or a small extension. In the binary case, the security of the keystream generation can be related, in the concrete security model, to the conjectured intractability of the MQ problem of solving a random system of m equations in n unknowns. We show that this security reduction can be extended to incorporate the key and IV setup and provide a security argument related to the whole stream cipher. We also briefly address software and hardware performance issues and show that if one is willing to pseudorandomly generate the systems of quadratic polynomials underlying the cipher, this leads to suprisingly inexpensive hardware implementations of QUAD.

Key words: MQ problem, stream cipher, provable security, Gröbner basis computation

1 Introduction

Symmetric ciphers can be broadly classified into two main families of encryption algorithms: block ciphers and stream ciphers. Unlike block ciphers, stream ciphers do not produce a key-dependent permutation over a large block space, but a key-dependent sequence of numbers over a small alphabet, typically the binary alphabet $\{0, 1\}$. To encrypt a plaintext sequence, each plaintext symbol is combined with the corresponding symbol of the keystream sequence by using a group operation, usually the exclusive or operation over $\{0, 1\}$. Nearly all stream ciphers specified recently use two inputs to generate a keystream sequence: a secret key and an additional parameter named initial value (IV) that is generally not secret. The use of IVs allows to derive several independent keystream sequences from one single key by resynchronizing the stream cipher each time with a new IV.

The current status of stream ciphers design is characterized by a considerable discrepancy between theory and practice.

On the theoretical side, seminal work by Shamir[31], Blum and Micali [7], Yao [33], Levin and Goldreich [21] in the early 80's produced the well founded theory of pseudo-random generators, which represents one of the major achievements in the area of provable security. A pseudo-random number generator (PRNG) can be viewed as an IV-less stream cipher. It expands a short seed, e.g. a key, into a larger bit string in such a way that if the input seed is secret and randomly drawn, then the resulting output is computationally indistinguishable from a perfect random sequence. The research effort on security proofs for PRNGs has not only led to remarkable generic results, e.g. the proof by Impagliazzo, Levin, Luby and Håstad [24] that a secure PRNG can be constructed based upon any one way function (OWF). It has also led to "provably secure" PRNG constructions exploiting the conjectured one-wayness of specific permutation or function f, which generally rely on the iteration of f and the extraction of a few bits at each iteration. The first construction of this type was introduced by Blum and Micali [7]. Its security reduction relates the security of the PRNG to the one-wayness of exponentiation modulo a prime number. The construction proposed by L. Blum, M. Blum and M. Shub [6] exploits the conjectured intractability of quadratic residuosity modulo Blum integers. Alexi, Chor, Goldreich and Schnorr proposed a construction with security that relies upon the RSA assumption. Impagliazzo and Naor [27] proposed a construction relying on the difficulty of the subset sum problem. More recently, some efficiency improvements were recently achieved, either by decreasing the state length as in Fisher and Stern's construction [16] based on the intractability of the syndrome decoding problem, or by increasing the number of bits extracted at each iteration of f as illustrated in Gennaro's construction based on the intractability of the discrete logarithm problem and Boneh, Halevi, Howgrave-Graham's construction [8] and in Steinfeld, Pieprzyk and Wang's construction [32] respectively based on the conjectured pseudo-randomness and one-wayness of RSA with small inputs. However, current provably secure PRNGs are still generally regarded as too complex and inefficient to provide really practical stream ciphers. The lack, for these various algorithms, of an extra IV parameter, represents an additional drawback.

On the practical side, extremely efficient stream ciphers have been proposed, which either allow like SCREAM [23], RC4 [30], SNOW 2.0 [14]and its UMTS variant SNOW 3G) much faster software encryption than existing block ciphers such as AES or require much lower computing resources for hardware implementations or both, like the GRAIN [26] and TRIVIUM [9] candidates to the ongoing European initiative eSTREAM [13]. However, the design of secure stream ciphers is not currently as well understood as the design of secure block ciphers. The state of the art of the cryptanalysis of stream ciphers has evolved significantly over the last ten years with the development of attack techniques such as algebraic attacks, fast correlation and linear masking attacks, resynchronization attacks against IV-dependent stream ciphers, and it turns out that many recent proposals still suffer from security weaknesses. This is illustrated by the fact that more than one third of the 34 candidate algorithms submitted to the eSTREAM stream ciphers evaluation project have already been shown to be insecure.

The main design objective of QUAD was to contribute to reducing the discrepancy between practical stream ciphers and provably secure PRNG constructions depicted above by specifying a practical stream cipher with provable security arguments. Instead of relying upon the conjectured intractability of number theoretical problems -e.g. the the factoring and discrete logarithm problems- like most provably secure PRNG constructions proposed so far, QUAD relies upon the conjectured intractability of the MQ problem of solving a multivariate system of m quadratic equations in n unknowns over a finite field GF(q), e.g. GF(2), which is known to be NP-hard and conjectured to be intractable in terms of average complexity even for extremely compact instances provided that the $\frac{m}{n}$ ratio is sufficiently close to 1. Thus QUAD belongs to the promising and fast expanding family of multivariate cryptographic algorithms. Moreover, unlike asymmetric multivariate algorithms relying upon the intractability of the MQ problem proposed so far, e.g. HFE or UOV, QUAD can be based on a random instance of MQ without any embedded trapdoor since QUAD is symmetric cipher and the computations performed at the sending and receiving side do not require any inversion of a MQ instance. In other words, QUAD's security relies more directly upon the intractability of the MQ problem than the one of asymmetric multivariate algorithms.

This paper is organized as follows. We first summarize the status of the MQ problem (Section 2) and recall basic security definitions in a concrete (non asymptotic) security model (Section 3). We then describe the QUAD algorithm (Section 4). We show that in the GF(2) case, the security of the QUAD's keystream generator can be provably related to the conjectured intractability of the MQ problem (Section 5). We show how to extend this proof, also in the GF(2) case, as to incorporate the key and IV setup to get a security reduction for the whole cipher (Section 6). Finally, we address software and hardware implementation issues (Section 7).

2 Multivariate quadratic systems

We consider a finite field GF(q). A multivariate quadratic equation (or equivalently a multivariate quadratic polynomial) in n variables over GF(q) is a polynomial of degree at most 2 in GF$(q)[x_1, \ldots, x_n]$ which can be written as

$$Q(x) = \sum_{1 \le i \le j \le n} \alpha_{i,j} x_i x_j + \sum_{1 \le i \le n} \beta_i x_i + \gamma,$$

with coefficients $\alpha_{i,j}$, β_i, and γ in GF(q). In the particular case $q = 2$, which is the one most often considered in the sequel, monomials $x_i x_i$ and x_i are equal.

It is easy to see that the set $\mathcal{Q}$ of multivariate quadratic polynomials in n variables is an N-dimensional vector space over $\mathrm{GF}(q)$, where $N = \frac{1}{2}n(n+3)+1$ if $q \neq 2$ and $N = \frac{1}{2}n(n+1)+1$ if $q = 2$. A basis of this vector space is given by the $N-1$ distinct monomial functions of degree one or two, and the non-null constant polynomial. Any element of $\mathcal{Q}$ can be represented by the N-tuple of its $\mathrm{GF}(q)$ coefficients in this basis. Throughout the rest of this paper, by a randomly chosen quadratic polynomial in n unknowns we mean the quadratic polynomial represented in the above basis by a uniformly and independently drawn N-tuple of $\mathrm{GF}(q)$ coefficients.

A multivariate quadratic system S of m quadratic equations in n variables over $\mathrm{GF}(q)$ consist of a set $(Q_1, \ldots, Q_m)$ of m quadratic polynomials in n variables over $\mathrm{GF}(q)$. In the sequel, by a randomly chosen system of m quadratic poynomials in n unknowns, we mean m independently and randomly chosen quadratic polynomials. Such a system is represented by mN coefficients uniformly and independently drawn from $\mathrm{GF}(q)$.

We define the problem of solving multivariate quadratic systems (MQ problem) as follows: given a multivariate quadratic system $S = (Q_1, \ldots, Q_m)$, of m quadratic equations over $\mathrm{GF}(q)$, find a value $x \in \mathrm{GF}(q)^n$, if any, such that $Q_i(x) = 0$ for all $1 \leq i \leq m$.

Depending on the respective values of n and m, instances of MQ can be either easy or very difficult to solve. For $m = 1$ the number of solutions is known [28] and it is quite easy to find one solution. When m is significantly smaller than n, that is for an underdefined quadratic system, finding a solution is much easier than the exhaustive search on the number of variables [10]. In the opposite situation of an overdefined system ($m > n$) containing nearly N linearly independent quadratic equations solving an MQ problem is easy by linearization. The total complexity is then only $O(n^6)$. However, for general values of m and n the MQ problem is known to be NP-hard, even when restricted to quadratic equations over $\mathrm{GF}(2)$ (see [18, 17]) or over any finite field (see [29]).

Moreover, what makes the MQ problem particularly well suited for cryptographic applications is that it is conjectured to be very difficult not only asymptotically and in worst case, but already for small suitably selected values of m and n and in terms of the average complexity of solving a random instance. The problem seems to be the most difficult when m is close to n. For $m = n$ and $q = 2$ the complexity of the best known solving algorithms is $2^{n - O(\sqrt(n))}$ and thus rather close to the 2^n complexity of exhaustive search, and totally out of reach of existing computers when n is larger than 100. Even when $q = 2$ and $m = kn$, where $k > 1$ is small enough compared with $\frac{n}{2}$, the best known algorithms XL [11] and improved variants of Buchbergers's Gröbner basis computation algorithm such as Faugère's F4 and F5 algorithms [15] are exponential in n for a randomly chosen quadratic system. Much research has been dedicated in the past years to the above problem [12, 11]. Bardet's Ph.D. thesis [2] provides an accurate analysis of the complexity of the most efficient algorithm computing Gröbner basis known to solve a random system of $m = kn$ equations in n unknowns.

3 The stream cipher QUAD

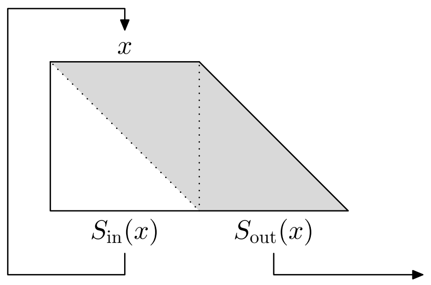
This section describes the stream cipher QUAD, which specification was first published in [5]. $S = (Q_1, \ldots, Q_{kn})$ denotes a multivariate quadratic system of kn randomly chosen equations in n variables over $GF(q)$, and S_0 and S_1 denote two (k times smaller) additional multivariate systems of n randomly chosen equations in n variables over $GF(q)$. S, S_0 and S_1 are fixed and publicly known. During the key and IV loading and the keystream generation, the internal register state is an n-tuple $x = (x_1, \ldots, x_n)$of $GF(q)$ values.

3.1 Keystream generation and encryption

The keystream generation process simply consists in iterating the three following steps in order to produce $(k-1)n$ $GF(q)$ keystream values at each iteration.

– **Step 1:** compute the kn-tuple of $GF(q)$ values $S(x) = (Q_1(x), \ldots, Q_{kn}(x))$ where x is the current value of the internal state;
– **Step 2:** output the keystream sequence $S_{out}(x) = (Q_{n+1}(x), \ldots, Q_{kn}(x))$ of $(k-1)n$ $GF(q)$ values
– **Step 3:** update the internal state x with the sequence of the n first generated $GF(q)$ values $S_{in}(x) = (Q_1(x), \ldots, Q_n(x))$

The maximal length of the keystream sequence that can be generated with a single (key,IV) pair is set to L $GF(q)$ symbols. In order to encrypt a plaintext of length $l \leq L$ $GF(q)$ symbols, each of the first l $GF(q)$ values of the keystream sequence is added (using the $GF(q)$ addition) with the corresponding plaintext value.



3.2 Key and IV setup

Before generating any keystream we need to initialize the internal state x, with the key K and the initialization vector IV, which are respectively represented by a sequence of $GF(q)$ elements of length $|K|$ and a binary sequence of $\{0,1\}$ values of length $|IV|$. We assume for the time being, for simplicity of the subsequent proofs that $|K|$ is chosen exactly equal to n.

The initialization is done as follows: we first set the internal state value x to the $GF(q)^n$ value K. Then for each of the $|IV|$ bits IV_1 to $IV_{|IV|}$ of the IV value, the internal state x is updated as follows: if $IV_i = 0$, x is replaced by the $GF(q)^n$ value $S_0(x)$; otherwise, x is replaced by the $GF(q)^n$ value $S_1(x)$. These $|IV|$ steps provide a key and IV dependent internal state value x. We then clock the cipher n additional times as described in section 3.1, but without outputting the keystream. After this preliminary runup phase, the keystream is generated as described in section 3.1.

4 Basic security notions

We first recall definitions of advantages for distinguishing a number generator from a perfect random generator and a function generator from a perfect random function generator, and the notions of Pseudo Random Number Generator (PRNG) and Pseudo Random Function (PRF). All the security definitions used throughout this paper relate to the concrete (non asymptotic) security model. In the sequel, when we state that a value u is randomly chosen in a set U, we implicitly mean that u is drawn according to the uniform law over U.

Single-query distinguisher for a number generator: let us consider a number generator $g : \{0,1\}^n \longrightarrow \{0,1\}^L$ with input and output lengths $L > n$, used to expand an n-bit secret random seed into an L-bit sequence. A distinguisher in time t for g is a probabilistic testing algorithm A which when input with an L-bit string outputs either 0 or 1 with time complexity at most t. We define the advantage of A for distinguishing g from a perfect random generator as

$$\mathbf{Adv}_g^{prng}(A) = \left|\Pr_{x\in\{0,1\}^n}(A(g(x)) = 1) - \Pr_{y\in\{0,1\}^L}(A(y) = 1)\right|,$$

where the probabilities are not only taken over the value of an unknown randomly chosen $x \in \{0,1\}^n$ (resp. of a randomly chosen $y \in \{0,1\}^L$), as explicitly stated in the above formula, but also over the random choices of the probabilistic algorithm A.

We define the advantage for distinguishing the function g in time t as

$$\mathbf{Adv}_g^{prng}(t) = \max_A\{\mathbf{Adv}_g^{prng}(A)\},$$

where the maximum is taken over all testing algorithms of time complexity at most t.

Pseudo Random Generator (PRNG): a function g is said to be a PRNG if $\mathbf{Adv}_g^{prng}(t)$ is negligible (for example less than 2^{-40}) for values of t strictly lower than a fixed threshold (for example 2^{80} or 2^{128}). The definition of a PRNG is therefore dependent upon thresholds reflecting the current perception of an acceptably secure number generator.

Distinguisher for a function generator: let us now consider a function generator, i.e. a family $F = \{f_K\}$ of $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions indexed by a key K randomly chosen from $\{0,1\}^k$. A distinguisher in time t with q queries for F is a probabilistic testing algorithm A^f capable to query an n-bit to m-bit oracle function f up to q times. Such an algorithm allows to distinguish a randomly chosen function f_K of F from a perfect random function f^* randomly chosen in the set $F^*_{n,m}$ of all $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions with a distinguishing advantage

$$\mathbf{Adv}_F^{prf}(A) = \left|\Pr(A^{f_K} = 1) - \Pr(A^{f^*} = 1)\right|,$$

where the probabilities are taken over $K \in \{0,1\}^k$ (resp $f^* \in F^*_{n,m}$) and over the random choices of A. We define the advantage for distinguishing the family F in time t with q queries as

$$\mathbf{Adv}_F^{prf}(t,q) = \max_A\{\mathbf{Adv}_F^{prf}(A)\},$$

where the maximum is taken over all testing algorithms A working in time at most t and capable to query an n-bit to m-bit oracle function up to q times.

Pseudo Random Function (PRF): a family of functions $F = \{f_K\}$ is said to be a PRF if $\mathbf{Adv}_F^{prf}(t,q)$ is negligible for values of t and q strictly lower than the respective threshold (for example 2^{80} or 2^{128} for t and 2^{40} for q).

5 Security of the keystream generation

We now give an outline of the security reduction relating, in the GF(2) case, the PRNG-indistiguishability of the keystream generation part of QUAD to the conjectured intractability of the MQ problem. This security reduction is expressed by Theorem 4 hereafter. The details of the proof of Theorem 4 are given in [5]. In this paper, we only describe the structure of this proof, which is divided in three parts.

5.1 Part 1: distinguishing the keystream allows to distinguish the output of a random quadratic function

In the first part (Theorem 1), we prove that if the L-bit keystream sequence associated with a known fixed or randomly chosen system S of $m = kn$ quadratic equations and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ is distinguishable from the L-bit output of a perfectly uniform generator, then for a known random quadratic system S of $m = kn$ equations and an unknown randomly chosen input value $x \in \{0,1\}^n$, $S(x)$ is distinguishable from a random kn bit word. Though we consider a randomly chosen system S because we need distinguishing properties related to a random system for the sequel, the property we prove would also hold if we considered a fixed system S. Our proof is inspired by the proof given in [22] that a similar result holds for the generator based on iteration of any fixed n-bit to m-bit function, where $m > n$, but provides a tighter bound for the advantage than [22].

Theorem 1. *Let $L = \lambda(k-1)n$ be the number of keystream bits produced in time λT_S using λ iterations of our construction. Suppose there is an algorithm A that distinguishes the L-bit keystream sequence associated with a known randomly chosen system S and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ from a random L-bit sequence in time T with advantage ϵ. Then there exists an algorithm B that for a randomly chosen S distinguishes $S(x)$ corresponding to an unknown random input x, from a random value of size kn in time $T' = T + \lambda T_S$ with advantage $\frac{\epsilon}{\lambda}$.*

5.2 Part 2: distinguishing the output of a random quadratic function allows to predict any linear function of its input

In the second part (Theorem 2), we prove that if for a known randomly chosen quadratic system S and an unknown randomly chosen x, there exists a distinguisher allowing to distinguish $S(x)$ from a random kn bit word such as the one considered in Theorem 1 above, then it can be converted into an algorithm allowing, for any n-bit to 1-bit quadratic function R, in particular any linear form R, to predict $R(x)$ for a randomly chosen n bit value x better than at random given $S(x)$.

Theorem 2. *Suppose there is an algorithm A that, given a randomly chosen known multivariate quadratic system S of kn equations in n unknowns, distinguishes $S(x)$, where x is an unknown random input value, from a random string of length kn with advantage at least ϵ and in time T. Then there is an algorithm B that, given a randomly chosen quadratic system S of kn equations in n unknowns, any n-bit to 1-bit quadratic form R, and $y = S(x)$ where x is a random input value, predicts $R(x)$ with success probability at least $\frac{1}{2} + \frac{\epsilon}{4}$ using at most $T' = T + 2T_S$ operations.*

5.3 Part 3: predicting any linear function of the input of a quadratic function allows to invert it

In the third part (Theorem 3), we show that if for a fixed or random quadratic system S and more generally any fixed or random n-bit to m-bit function f there exists a predictor such as the one considered in the former theorem, i.e. a predictor allowing, given an n-bit to 1-bit linear form R, to predict $R(x)$ with a success probability (over all S and x values) strictly larger than $\frac{1}{2}$, then a preimage of $S(x)$ (resp. f(x)) can be efficiently computed, so that S (resp f) is not one way. This part is essentially a proof of Goldreich-Levin's theorem ([21]), in which a fast Walsh transform computation is used to get a tighter reduction. In order to proof Theorem 3, which relates to the computation, given the image $S(x)$ or $f(x)$ for a random unknown value x and a random system S, of a list containing x, we first establish a lemma representing the technical core of the proof, in which a fixed (unknown) value of x is considered. Our proofs are inspired by the simplified treatment of the original Goldreich-Levin proofs developed by Rackoff, Goldreich in [19] and Bellare in [3], and also by the proofs provided by Håstad and Näslund in [25].

Lemma 1. *Let us denote by x a fixed unknown n-bit value and denote by f a fixed n-bit to m-bit function. Suppose there exists an algorithm B that given the value of $f(x)$ allows to predict the value of any linear equation R over n unknowns with probability $\frac{1}{2}+\epsilon$ over R, using at most T operations. Then there exists an algorithm C, which given $f(x)$ produces in time at most T' a list of at most $4n^2\epsilon^{-2}$ values such that the probability that x appears in this list is at least* 1/2.

$$T' = \frac{2n^2}{\epsilon^2}\left(T + \log\left(\frac{2n}{\epsilon^2}\right) + 2\right) + \frac{2n}{\epsilon^2}T_f$$

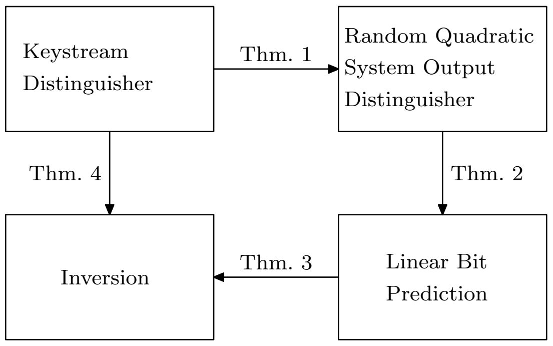
This Lemma applies to a fixed x and a fixed system S (or a fixed n-bit to m-bit function f). However, the success probability of the predictor of Theorem 2 is taken over all $(x,\ S)$ pairs for any linear form R. Consequently, we need a theorem allowing us to exploit the existence of such a predictor to show the applicability of the lemma to a non-negligible fraction of (x, S) pairs.

Theorem 3. *Suppose there exists an algorithm B, that given a randomly chosen quadratic system S of m quadratic equations, a randomly chosen n-bit to 1-bit quadratic form R and the image $S(x)$ of a randomly chosen (unknown) n-bit value x, predicts the value of $R(x)$ with probability at least $\frac{1}{2}+\epsilon$ over all possible (x, S, R) triplets using T operations. Then there is an algorithm C, which given the image $S(x)$ of a randomly chosen (unknown) n-bit value x produces a preimage of $S(x)$ with probability at least $\epsilon/2$ (over all possible values of x and S) in time T'.*

$$T' = \frac{8n^2}{\epsilon^2}\left(T + \log\left(\frac{8n}{\epsilon^2}\right) + 2\right) + \frac{8n}{\epsilon^2}T_f$$

5.4 Security proof for the keystream generation

Now it is easy to see that if we sequentially apply theorems 1, 2, and 3, we obtain the following reduction theorem, which states that if, for a random system and a random initial value, the L-bit keystream sequence was distinguishable from a random L-bit sequence then there would exist an efficient algorithm allowing to find a preimage of the image of a random n-bit input value by a random quadratic n-bit to m-bit system, which for suitably chosen values of n would contradict the assumptions made in Section 2 on the difficulty of solving MQ.



Theorem 4. *Let $L = \lambda(k-1)n$ be the number of keystream bits produced by in time λT_S using λ iterations of our construction. Suppose there exists an algorithm A that distinguishes the L-bit keystream sequence associated with a known randomly chosen system S and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ from a random L-bit sequence in time T with advantage ϵ. Then there exists an algorithm C, which given the image $S(x)$ of a randomly chosen (unknown) n-bit value x by a randomly chosen n-bit to m-bit quadratic system S produces a preimage of $S(x)$ with probability at least $\frac{\epsilon}{2^3\lambda}$ over all possible values of x and S in time upper bounded by T'.*

$$T' = \frac{2^7 n^2 \lambda^2}{\epsilon^2}\left(T + (\lambda+2)T_S + \log\left(\frac{2^7 n\lambda^2}{\epsilon^2}\right) + 2\right) + \frac{2^7 n\lambda^2}{\epsilon^2} T_S$$

Theorem 4 above relates to the keystream generation part of QUAD only, not to the key and IV setup computation for deriving the initial state. Moreover it does not guarantee the strength of a particular instance of QUAD associated with a fixed system S but (informally) it shows that for suitably chosen parameter values if MQ is intractable then most instances of QUAD are secure.

5.5 Specifying Parameter Values for QUAD

We now propose concrete parameters n, k, and L for our construction. We still restrict ourselves to the $GF(2)$ case. We want to ensure a security level of at least 2^{80}. More precisely we want Theorem 4 to ensure that if for a random system and a random initial internal state value at the beginning of the keystream generation there exists a testing algorithm that allows us to distinguish an L-bit keystream produced by QUAD from a uniformly drawn keystream sequence with an advantage of more than $\epsilon = \frac{1}{100}$ in time less than $T = 2^{80}$, this would imply the existence of an inversion algorithm of non negligible success probability $\epsilon' = \frac{\epsilon}{2^3\lambda}$ allowing, given a random n-bit to kn-bit system of quadratic equations and the $S(x)$ image by S of a random input value x, to find a preimage by S of $S(x)$ in time T' lower by a substantial factor, say $\frac{1}{\epsilon'}$, than the best known inversion algorithms for the MQ problem, and thus result in the existence of a large set of weak instances of MQ.

Depending on the intended application of the pseudorandom number generator, the maximum keystream length L can vary from a few hundreds bits for a mobile phone application to up to 2^{40} bits. Consequently the allowed parameter values for n and k will also vary, since it is much more demanding to get a security argument for $L = 2^{40}$ bits than for $L = 1000$ bits. We will however retain the latter value $L = 2^{40}$ for a first estimate of the corresponding required value of n.

In her thesis, Magali Bardet [2] shows that the best Groebner basis algorithm to solve a system of kn equations in k unknowns has (in the case of a regular system) a complexity of $T(k,n) = \left(\binom{n+1}{D}\right)^{2.37}$, where D is close to $\left(-k + \frac{1}{2} + \frac{1}{2}\sqrt{2k^2 - 10k - 1 + 2(k+2)\sqrt{k(k+2)}}\right) n$. To obtain a contradiction, we need to have T' lower than $\epsilon' T(k,n)$. For $k = 2$ and with the previous values of $L = 2^{40}$, $T = 2^{80}$ and $\epsilon = \frac{1}{100}$, we get $\epsilon' = 2^{-42}$ and we need to have n greater than 350. For $n = 256$ and $k = 2$, we only get a contradiction if we produce less than $L = 2^{22} = 4$ Mbits of keystream.

Parameter values recommended in practice: for QUAD over GF(2), we recommend in practice an internal state length of $n = 160$ bits and an expansion factor k of 2 and a maximum keystream length $L = 2^{40}$. For such n, k and L values, the former concrete security reduction is not applicable, i.e. we do not get a contradiction as for the former parameter values. However our proof reduction is not optimal, and we conjecture that these parameter values suffice to provide the desired security level of at least 2^{80}.

6 Extending the security proof to the Key and IV setup

The security proof of the former section only relates to the keystream generation part of QUAD. We now extend this proof to include the key and IV setup. Our aim is to relate the indistinguishability of the QUAD cipher, more precisely of the family of IV to keystream functions indexed by the key K associated with QUAD, to the conjectured intractability of the MQ problem. For that purpose, we view QUAD as the composition of two functions, and provide security proofs for these functions.

– A keyed **initial state derivation function**, which consists of the initial phase of the key and IV setup, i.e. the derivation of the initial state before the runup phase. This part can be viewed as a family of IV to initial state functions indexed by the key K. We will show that for suitably chosen parameter sizes, this family of functions can be expected to be a PRF.
– An unkeyed **initial state to keystream function**, which consists of the runup phase of the key and IV setup followed by the keystream generation. We will show that for suitably chosen parameter sizes, this function can be expected to be a PRNG.

A simple composition theorem (stating essentially that the composition of a PRF and a PRNG with fitting output and input lengths is a PRF) allows then to derive a security reduction for the whole cipher.

6.1 PRF-indistiguishability of the initial state derivation function of QUAD

The key observation allowing to establish that if the MQ problem is intractable then the initial state derivation function of QUAD is a secure is that this function results from applying the Tree Based Construction introduced by Goldreich, Goldwasser, and Micali in [20] to the quadratic n-bit to $2n$-bit function $S' = (S_0, S_1)$. Indeed, the IV bits determine a path leading from the key to the initial state in the binary tree induced by S' and thus plays exactly the role of the input bits in the Tree Based Construction.

The Tree Based Construction is a generic construction allowing to derive a PRF from a PRNG. It can be defined as follows. Let us consider a PRNG $g : \{0,1\}^m \longrightarrow \{0,1\}^{2m}$ and let us denote the $2m$-bit image of $y \in \{0,1\}^m$ by $g(y) = z_0 z_1 \dots z_{2m-1}$. We derive from g two m-bit to m-bit functions $g_0 : y \in \{0,1\}^m \longmapsto z_0, \dots, z_{m-1}$ and $g_1 : y \in \{0,1\}^m \longmapsto z_m, \dots, z_{2m-1}$.

The PRF F^g is the family of functions $\{f_y\}_{y \in \{0,1\}^m}$ where

$$\begin{aligned} f_y : \{0,1\}^n &\longrightarrow \{0,1\}^m \\ (x_1, x_2, \dots, x_n) &\longmapsto f_y(x_1, x_2, \dots, x_n) = g_{x_n} \circ g_{x_{n-1}} \dots \circ g_{x_1}(y) \end{aligned}$$

This construction is illustrated on Figure 4.

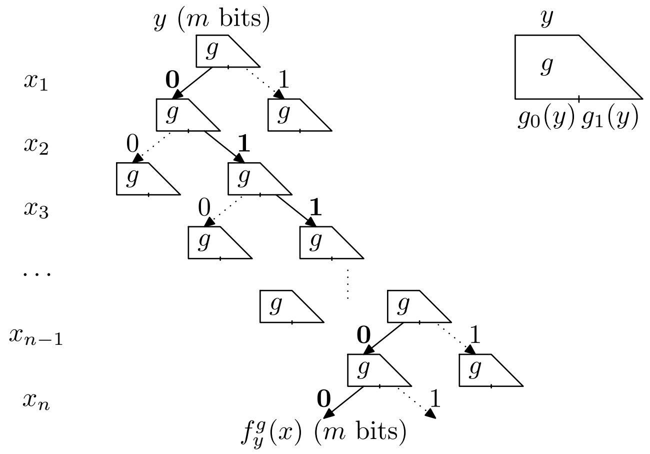


Fig. 1. Tree Based Construction

The following Theorem relates the PRF-advantage for distinguishing F^g to the PRNG-advantage for distinguishing g. The proof is essentially the same as the security proof for the Tree Based Construction given by Goldreich in [19], up to the fact that we consider the concrete security model instead of the asymptotic polynomial time indistinguishability model.

Theorem 5. *Let $g : \{0,1\}^m \longrightarrow \{0,1\}^{2m}$ be a number generator which generates $2m$ outputs bits in time T_g^{2m} and let $F^g = \{f_y\}_{y \in \{0,1\}^m}$ be the family of n-bit to m-bit functions derived from g by the Tree Based Construction. The (t, q) PRF advantage of F^g is related to the single-query PRNG advantage of g by the following inequality*

$$\mathbf{Adv}_{F^g}^{prf}(t, q) \leq nq\mathbf{Adv}_g^{prng}(t + q(n+1)T_g^{2m}).$$

The application of Theorem 5 to the initial state derivation of QUAD is straightforward. We denote by $g^{S'} : \{0,1\}^n \longrightarrow \{0,1\}^{2n}$ the n-bit to $2n$-bit function associated with S', and by $F^{g^{S'}}$ the resulting family of initial state derivation functions. Theorem 5 allows to relate the PRF-advantage for distinguishing $F^{g^{S'}}$ to the PRNG-advantage for distinguising $g^{S'}$ by the inequality

$$\mathbf{Adv}^{prf}_{F^{g^{S'}}}(t,q) \leq 2q\mathbf{Adv}^{prng}_{g^{S'}}(t + q(|IV|+2)T^{2n}_{g^{S'}})$$

Moreover, Theorem 4 above allows to relate the PRNG-advantage for distinguish $g^{S'}$ to the hardness of inverting MQ.

6.2 PRNG-indistinguishability of the initial state to keystream function

Let us denote by $g^S : \{0,1\}^n \longrightarrow \{0,1\}^L$ the keystream generation function induced by the iteration of the quadratic function S. The security of the number generator g_{real} which starts by running n clocks like g^S without producing any keystream to reflect the runup of QUAD and then produces L bits of keystream in the same way as g^S is related to the security of $\tilde{g}^S : \{0,1\}^n \longrightarrow \{0,1\}^{L+(k-1)n^2}$ which iterates S to produce $L + (k-1)n^2$ bits, since g_{real} produces the same keystream as $\tilde{g}^S$ up to the fact that the first $(k-1)n^2$ bits of $g^{S'}$ are discarded. Consequently a distinguisher for g_{real} is also a distinguisher for $\tilde{g}^S$. Thus the advantage of g_{real} is upper-bounded by the advantage of $\tilde{g}^S$.

$$\mathbf{Adv}^{prng}_{g_{real}}(t) \leq \mathbf{Adv}^{prng}_{\tilde{g}^S}(t + T^{L+(k-1)n^2}_{\tilde{g}^S})$$

6.3 PRF-indistinguishability of the whole cipher

Now a simlpe composition theorem (Theorem 6 hereafter) allows to derive from the two former results a security reduction related to the whole cipher (Theorem 7 hereafter). We define the composition G of a family of function F and a function g, and relate the PRF-indistinguishability of G to the PRF-indistinguishability of F and the PRNG-indistinguishability of g.

Definition 1. *The composition $G = g \circ F$ of an n-bit to m-bit family of functions $F = \{f_K\}$ and of an m-bit to L-bit function g is the n-bit to L-bit family of functions*

$$G = \{g \circ f_K\}.$$

Theorem 6. *Let us consider $F = \{f_K\}$ where $f_K : \{0,1\}^n \longrightarrow \{0,1\}^m$ a functions family and $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ a number generator that produces L bits in time T^L_g. The (t,q) advantage of $G = g \circ F = \{g \circ f_K\}$ can be upper bounded as follows*

$$\mathbf{Adv}^{prf}_G(t,q) \leq \mathbf{Adv}^{prf}_F(t + qT^L_g) + q\mathbf{Adv}^{prng}_g(t + qT^L_g).$$

The stream cipher QUAD results from the composition of the function family $F^{g^{S'}}$ and the number generator g_{real}. Due to the composition Theorem 6, the security of QUAD can be related to the security of $F^{g^{S'}}$ and g_{real} which can in turn, as established in the former sections, be related to the security $g^{S'}$ and $\tilde{g}^S$. We get the inequality:

$$\mathbf{Adv}^{prf}_{\text{QUAD}}(t,q) \leq 2q\mathbf{Adv}^{prng}_{g^{S'}}(t+q(|IV|+3)T^L_{g^{S'}})+q\mathbf{Adv}^{prng}_{\tilde{g}^S}(t+qT^L_{\tilde{g}^S}+T^{L+(k-1)n^2}_{\tilde{g}^S})$$

The initial state derivation and initial state to keystream functions of QUAD are based on the iteration of a randomly chosen quadratic system of km equations in m variables (with $k=2$ for S'). If S' and S are equal (or S' consists of the $2n$ first polynomials of S) we have:

$$\mathbf{Adv}^{prf}_{\text{QUAD}}(t,q) \leq 3q\mathbf{Adv}^{prng}_{g^S}(t+q(|IV|+3)T^{L+(k-1)n^2}_{\tilde{g}^S})$$

We can now use the results of Theorem 4 to relate the security of the whole stream cipher QUAD to the difficulty of the MQ problem, i.e. show that if there exists an adversary capable to distinguish QUAD from a perfect random function in time t with q queries then there exists a MQ solver. We estimate the time to compute a quadratic equation in n variables to n^2 and the time to compute a system of kn equations in n unknowns to kn^3.

Theorem 7. *Let us denote $\lambda = \frac{L+(k-1)n^2}{(k-1)n}$. Suppose there exists an algorithm A which distinguishes the stream cipher QUAD producing L keystream bits for each of the $2^{|IV|}$ IVs from a perfect random function in time T, with q queries and a PRF-advantage ϵ. Then there exists an algorithm B which given the image $S(x)$ of a randomly (unknown) n-bit value x by a randomly chosen n-bit to kn-bit quadratic system S produces a preimage of $S(x)$ with probability at least $\frac{\epsilon}{3\cdot 2^3 q\lambda}$ over all possible values of x and S in time upper bounded by T'.*

$$T' = \frac{9\cdot 2^7 n^2\lambda^2 q^2}{\epsilon^2}\left(T + q(|IV|+2)\lambda n^3 + (\lambda+4)kn^2 + \log\left(\frac{9\cdot 2^7 n\lambda^2 q^2}{\epsilon^2}\right) + 2\right)$$

Now we have extended the security proof of QUAD to include the key and IV setup we can, as done after the security proof of the keystream generation, propose parameter values for n, k, L, $|K|$ and $|IV|$ allowing to get a concrete security reduction for whole stream cipher. We restrict ourselves to the $GF(2)$ case. We want to ensure a security level of at least 2^{80}. More precisely, we want Theorem 7 to ensure that the existence of an algorithm allowing, for a randomly chosen system S, to distinguish the IV to keystream function induced by the stream cipher QUAD and a random key from a random function with q queries and an advantage of more than $\epsilon = \frac{1}{100}$ in time less than $T = 2^{80}$ would imply the existence of an inversion algorithm of non negligible success probability $\epsilon' = \frac{\epsilon}{3\cdot 2^3\lambda q}$ allowing, given a random n-bit to kn-bit system of quadratic equations and the image $S(x)$ of a random input value x, to find a preimage of $S(x)$ by S in time T' substantially lower, by a factor of more than ϵ', than the best known inversion algorithms for the MQ problem, and thus the existence of a large set of weak instances of MQ.

For $k = 2$ and with the previous values of $L = 2^{40}$, $q = 2^{40}$, $T = 2^{80}$ and $\epsilon = \frac{1}{100}$, we get $\epsilon' = 2^{-78}$ and we need to have n greater than 760. For $n = 512$ and $k = 2$, we only get a contradiction if we produce less than $L = 2^{21}$ bits of keystream and allow up to $q = 2^{30}$ queries. These values of n are higher than those for the keystream generation. However our proofs are not perfectly tight and 760 bits is still quite low compared to the size stream ciphers based on discrete log or RSA would require.

7 Software and Hardware Implementation of QUAD

7.1 Software Implementation

Implementing QUAD in software essentially amounts to computing a system of quadratic functions in n variables over GF(q). This holds for the the key and IV setup and the runup and keystream generation phases of QUAD, the main differences between the two phases being that the number of quadratic functions one has to compute at each iteration are n and $m > n$ respectively. There are two main steps in the computation of a system of m quadratic functions $Q_1, \cdots, Q_m$ in n variables $x_1, \cdots, x_n$: firstly a monomials generation step which consists of computing the $N = \frac{1}{2}n(n+3)+1$ (($q \neq 2$) case) or $N = \frac{1}{2}n(n+1)+1$ monomials of degree 0, 1 or 2. Secondly a polynomials computation step which can be viewed as the computation of the product of the vector of the N monomials produced at the first step by the $m \times N$ matrix Q which rows are the coefficients of the m quadratic functions $Q_1, \cdots, Q_m$. Various techniques allowing to efficiently implement these two steps are described in [4]. In the case of a system over a small extension of GF(2), e.g. GF(2^4), the use of bitslicing techniques allows to speed up the first step by computing several monomials in parallel, whereas the use of lookup tables containing, for each column vector of Q and each scalar coefficient $a \in$ GF(q), the product of the column vector by a allows to speed up the second step.

name	**vendor**	**processor**	**frequency**	**L2 cache**
M1	Intel	Pentium 4	2505 MHz	512 kB
M2	Intel	Pentium M	1862 MHz	2048 kB
M3	Intel	Xeon	2784 MHz	512 kB
M4	AMD	Opteron	2197 MHz	1024 kB
M5	AMD	AMD64	1790 MHz	512 kB
M6	AMD	Athlon XP	2162 MHz	512 kB
M7	Power PC	G3	900 MHz	512 kB

Implementations of QUAD in C were produced for the two sets of parameters described hereafter, and a modified version of the eSTREAM Testing Framework made by C. de Cannière [13] was used to evaluate the performance of these implementations on various platforms listed in the table below when compiled with different compilers and compiling options. We mostly used compilers `gcc-4`, `gcc-3.4`, `gcc-3.3`, and `gcc-2.95`, although Intel's `icc` compiler was also supported.

The two parameters sets we considered are the following:

- The GF(2) version of QUAD with a 160-bit state in which a system of 320 equations in 160 variables is iterated recommended in Section 6. Although $n = 160$ is not enough in order for the security reduction of QUAD to give any formal reduction argument, i.e. any contradiction with the conjectured intractability of MQ, we believe that this is in practice a rather conservative instance of QUAD.
- The GF(16) version of QUAD with a 160-bit state in which a system of 80 equations in 40 variables is iterated. Though there is no evidence so far that existing methods result in an attack of complexity less that 2^{80} against the keystream generator, the underlying MQ problem of solving a system of 80 equations in 40 variables can be solved in substantially less than 2^{80} GF(16) operations. Therefore, this is a much less conservative version of QUAD than the former one that we do not recommend for use in applications with strong security requirements. Although it does not make much sense to compare the performance of two instances which do not offer the same security level, the higher throughput achieved with this version suggests that one may expect some performance improvements when implementing QUAD on a small extension of GF(2) rather than GF(2).

The associated performance figures (in cycles per byte) are given in the two tables hereafter. The orders of magnitudes of the encryption speeds obtained for fastest implementations of the GF(2) instance and the GF(16) instance are 8 Mbit/s and 24 Mbit/s.

Table 1. GF(2) case: $n = 160$, $t = 2$ - speed in cycles/byte

version	M1	M2	M3	M4	M5	M6	M7
32 bit	7057	3746	4600	**2930**	3205	4866	4983
64 bit				**2081**	2636		

Table 2. GF(16) case: $n = 40$, $t = 2$ - speed in cycles/byte

version	M1	M2	M3	M4	M5	M6	M7
32 bit	1906	1204	1849	1003	990	1257	**874**
64 bit				**745**	885		

7.2 Hardware Performance

It might seem at first glance that QUAD is not well fit for hardware implementations because it is impractical to manage a large random system of quadratic equations in hardware. We show in this section that however, if one is willing to generate the fixed multivariate quadratic function S iterated by QUAD pseudo-randomly rather than randomly by means of a simple non linear number generator, this results in surprisingly good hardware performance: about 3500 Gate Equivalents (GE) for the smallest implementation reported here. Though one can argue that the security reduction relating the indistinguishability of the QUAD output to the intractability of a random MQ instance can no longer be invoked in such a setting and that moreover we are considering smaller parameter sizes than those needed to get a strong formal security argument, we think that the existence of a strong link between the security of QUAD and the one of the underlying MQ problem still provides a partial security argument in this setting. We implemented binary version of QUAD with state lengths of $n = 128$, 160 and 256 bits (for an intended security level of approximately $2^{\frac{n}{2}}$ on a Xilinx Virtex4 FPGA. For each state length, we developed two main implementations realizing distinct trade-offs between area and throughput.

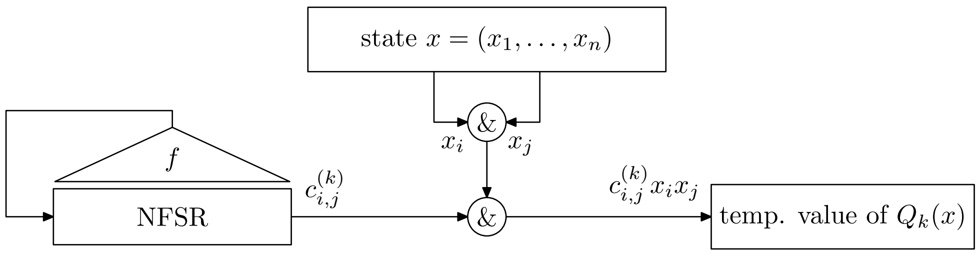


Fig. 2. The low area design $QUAD^{low}$

In the first implementation, named $QUAD^{low}$, area is minimized but the throughput is rather low. The second implementation, named $QUAD^{medium}$, achieves a much higher throughput at the expense of a moderate area increase. In the $QUAD^{low}$ implementation, we performed all binary operations sequentially in order to save area, as depicted in Figure 2. The implementation involves two main components. The first component is a linear feedback shift register (NFSR) which provides the sequence of binary coefficients for $Q_1, \cdots, Q_m$, where $m = 2n$. In our implementation, we used one of the NFSRs of the Achterbahn stream cipher proposal, of length 31 and period $2^{31} - 1$. The second component contains the current state x and produces the associated sequence of momomials $x_i x_j$. At each step, the AND product of the current coefficient and the current monomial is computed, and accumulated in a temporary output value. At the end of the computation, the obtained final output value is partly fed back to the state memory, and partly output as keystream bits.

Table 3. Low area implementation

Version	**128 bits**	**160 bits**	**256 bits**
Flip/Flops	66	68	68
4 input LUTs	153	169	181
Slices	85	92	97
Gate Equiv. (GE)	2961	3694	4611
Max. Freq. (MHz)	267	244	243
Throughput (Kbps)	16.1	9.5	3.7

Table 4. Medium area implementation with an improved throughput/area ratio

Version	**128 bits**	**160 bits**	**256 bits**
Flip/Flops	350	418	613
4 input LUTs	781	970	1471
Slices	406	509	763
GE	8117	10184	14959
Max. Freq. (MHz)	262	269	260
Throughput (Mbps)	4.1	3.3	2.0

In order to improve the throughput, the $QUAD^{medium}$, implementation simultaneously computes for each pair $(i, j) \in \{1, \cdots, n\} \times \{1, \cdots, m\}$, the $2n$-bit word of the coefficients of monomial $x_i x_j$ in $Q_1, \cdots, Q_m$. For that purpose, the NFSR component of $QUAD^{low}$ was replaced by a more expensive finite state machine (FSM), in which we used the S-boxes of Serpent [1] to produce $2n$ bits at each iteration. Tables 3 and 4 provide detailed performance figures for $QUAD^{low}$, and $QUAD^{medium}$. One can see that for the 160-bit versions, the orders of magnitude of the gate counts are 3500 GE and 10000 GE, whereas the obtained throughputs are approximately 10 Kbit/s and 3.3 Mbit/s.

8 Conclusion

QUAD is a practical stream cipher whose security is provably related, for suitable parameter values, to the conjectured intractability of the MQ problem. QUAD seems well suited for three main kinds of environments:

– **software platforms**, e.g. on PCs, for applications where the use of a cipher of unusually strong security arguments matters and where a throughput of a few Mbit/s is sufficient;

– **embedded devices with hardware encryption capabilities**, e.g. mobile stations. The area-throughput trade-off given by $QUAD^{medium}$ seems best suited for this use case. Due to the fact that for suitable parameter values, the quadratic function iterated in QUAD is expected to be strongly one-way, modes of operation of QUAD allowing to provide other security functionalities than mere encryption, in particular authentication and key agreement, are easy to define.
– **lightweight devices with highly limited computation capabilities such as RFID tags**. On such environments, QUAD can be used to provide encryption and even more advanced security functionnalities such as untraceable identification with forward security. The most compact hardware implementation of QUAD seems to be best suited for this environment.

References

1. Rosss Anderson, Eli Biham, and Lars Ramkilde Knudsen. Serpent: A flexible block cipher with maximum assurance. In *Proceedings of the First Advanced Encryption Standard Conference*, 1998.
2. Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université Paris VI, 2004.
3. Mihir Bellare. The Goldreich-Levin Theorem. `http://www-cse.ucsd.edu/users/mihir/courses.html`, 1999.
4. Côme Berbain, Olivier Billet, and Henri Gilbert. Efficient implementations of multivariate quadratic systems. In Selected Areas in Cryptography – SAC 2006, To appear in Lecture Notes in Computer Science. Springer-Verlag, 2006.
5. Côme Berbain, Henri Gilbert, and Jacques Patarin. QUAD: a Practical Stream Cipher with Provable Security. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, Lecture Notes in Computer Science. Springer-Verlag, 2006.
6. Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.
7. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
8. Dan Boneh, Shai Halevi, and Nick Howgrave-Graham. The modular inversion hidden number problem. In *ASIACRYPT*, pages 36–51, 2001.
9. Christophe De Cannière and Bart Preneel. Trivium: Specifications. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2005. Available at `http://www.ecrypt.eu.org/stream`.
10. Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier. Solving underdefined systems of multivariate quadratic equations. In *Public Key Cryptography*, pages 211–227, 2002.
11. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer-Verlag, 2000.
12. Nicolas Courtois and Jacques Patarin. About the XL Algorithm over $GF(2)$. In Marc Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 141–157. Springer-Verlag, 2003.
13. ECRYPT. eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932. Available at `http://www.ecrypt.eu.org/stream/`, Accessed September 29, 2005, 2005.

14. Patrik Ekdahl and Thomas Johansson. A new version of the stream cipher SNOW. In Kaisa Nyberg and Howard M. Heys, editors, *Proceedings of Selected Areas in Cryptography – SAC'02*, number 2595, 2002.
15. Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, Makoto Sugita, and Gwénolé Ars. Comparison Between XL and Grbner Basis Algorithms. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 338–353. Springer-Verlag, 2004.
16. Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *EUROCRYPT*, pages 245–255, 1996.
17. Aviezri S. Fraenkel and Yaacov Yesha. Complexity of solving algebraic equations. *Inf. Process. Lett.*, 10(4/5):178–179, 1980.
18. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, chapter 7.2 Algebraic Equations over $GF(2)$. W H Freeman & Co, 1979.
19. Oded Goldreich. *The Foundations of Cryptography - Volume 1.* Cambridge University Press, 2001.
20. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *J. ACM*, 33(4):792–807, 1986.
21. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In D. S. Johnson, editor, *21th ACM Symposium on Theory of Computing – STOC '89*, pages 25–32. ACM Press, 1989.
22. Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. Available at `http://www-cse.ucsd.edu/users/mihir/courses.html`, 2001.
23. Shai Halevi, Don Coppersmith, and Charanjit S. Jutla. Scream: A software-efficient stream cipher. In Joan Daemen and Vincent Rijmen, editors, *Proceedings of Fast Software Encryption – FSE'02*, number 2365, pages 195–209, 2002.
24. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
25. Johan Håstad and Mats Näslund. BMGL: Synchronous key-stream henerator with provable security. submitted to Nessie Project, 2000.
26. M. Hell, T. Johansson, and W. Meier. Grain - A Stream Cipher for Constrained Environments. ECRYPT Stream Cipher Project Report 2005/001, 2005. `http://www.ecrypt.eu.org/stream`.
27. Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.
28. Rudolf Lidl and Haradl Niederreiter. *Finite Fields.* Cambride University Press, 1997.
29. Jacques Patarin and Louis Goubin. Asymmetric cryptography with s-boxes. In *ICICS*, pages 369–380, 1997.
30. Ronald Rivest. The RC4 enycryption algorithm. RSA Security Inc., March 1992.
31. Adi Shamir. On the Generation of Cryptographically Strong Pseudo-Random Sequences. In *ICALP*, pages 544–550, 1981.
32. Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. On the provable security of an efficient rsa-based pseudorandom generator. In *ASIACRYPT*, pages 194–209, 2006.
33. Andrew Yao. Theory and Applications of Trapdoor Function. In *Foundations of Cryptography FOCS 1982*, 1982.