

© 2016 Kartik Palani

SCALABLE AUTHENTICATION FOR CONSUMER-SIDE SMART  
GRID INTERNET OF THINGS

BY

KARTIK PALANI

THESIS

Submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Electrical and Computer Engineering  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Adviser:

Professor David M. Nicol

# ABSTRACT

As computing becomes ubiquitous, there is an increased interaction between the computing devices. The transformation of the Internet of Computers (IoC) into the Internet of Things (IoT) elicits the need to revisit security schemes and to ask questions about their scalability. The need for such questions is further motivated by the direct connection between these constrained devices and critical infrastructure like the smart grid.

In this thesis we explore the scalability of smart grid consumer-side IoT. We look at how the extremely connected nature of IoT makes vulnerability blooms much easier and ask if existing solutions like X.509 certificates scale to large device populations. We also look at what information needs to be exchanged between devices in order to provide strong and secure authentication.

The evaluation shows that existing schemes need to be given a second look and in the rush to deploy solutions one must not plaster such schemes onto the large population of devices that have been envisioned as a part of the new age of computers. In particular we find that certificate schemes like certificate revocation do not scale to large populations. We attempt to solve the problem by a HMAC-based scheme.

*To my family, for their love and support.*

# TABLE OF CONTENTS

LIST OF ABBREVIATIONS . . . . .	v
CHAPTER 1 INTRODUCTION . . . . .	1
CHAPTER 2 BACKGROUND . . . . .	4
2.1 Authentication in Large-Scale Distributed Systems . . . . .	5
2.2 Authentication in the New Age of Computers . . . . .	9
CHAPTER 3 MOTIVATION . . . . .	15
3.1 IoC to IoT . . . . .	15
3.2 Approaching the Problem . . . . .	17
3.3 Measuring Security Health . . . . .	17
3.4 Modeling Health in the IoT . . . . .	21
3.5 Initial Results . . . . .	24
CHAPTER 4 EVALUATING THE SCALABILITY OF REVO- CATION SCHEMES FOR THE INTERNET OF THINGS . . . . .	27
4.1 Problem Description . . . . .	28
4.2 Design Description . . . . .	31
4.3 Modeling Approach . . . . .	32
4.4 Results . . . . .	41
CHAPTER 5 SCALABLE IDENTITY FOR SMART GRID IOT . . . . .	46
5.1 Proposed Architectures . . . . .	47
5.2 Results and Discussion . . . . .	53
CHAPTER 6 CONCLUSION AND FUTURE WORK . . . . .	57
REFERENCES . . . . .	59

# LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
AMI	Advanced Metering Infrastructure
BGP	Border Gateway Protocol
BLE	Bluetooth Low Energy
CA	Certificate Authority
CRL	Certificate Revocation List
CRT	Certificate Revocation Tree
CVE	Common Vulnerabilities and Exposures
DSA	Digital Signature Algorithm
ECU	Engine Control Unit
GSM	Global System for Mobile Communications
HMAC	keyed-Hash Message Authentication Code
ICS	Industrial Control Systems
IoT	Internet of Things
IoC	Internet of Computers
KDC	Key Distribution Center
KDF	Key Derivation Function
MAC	Message Authentication Code
MQTT	Message Queuing Telemetry Transport
OCSP	Online Certificate Status Protocol

PKI	Public Key Infrastructure
ROM	Read Only Memory
SDSI	Simple Distributed Security Infrastructure
SAN	Stochastic Activity Network
SCADA	Supervisory Control And Data Acquisition
SHA	Secure Hash Algorithm
SIM	Subscriber Identity Module
SPN	Stochastic Petri Net
SSL	Secure Sockets Layer
STNR	Server To Node Ratio
TLS	Transport Layer Security
TV	Television
USB	Universal Serial Bus

# CHAPTER 1

## INTRODUCTION

In standard visions of the Internet of Things (IoT), there are a massive number of things talking to each other. For this talking to be meaningful, the listeners need to know who the talkers are.

Was it really my smartphone app that just asked my front door to unlock and the heat to turn on (e.g. [1])? Was it really my car’s ECU that just told my car’s brakes to engage—or was it an impostor (e.g., [2])? Was it really Google Calendar that just asked my smart refrigerator for my password—or was it an impostor (e.g., [3])? Was it really my washing machine that just told my utility company that the machine is using a less power-hungry washing algorithm? Thanks to the permeable nature of networked communication, impersonation is always a concern.

For a representative example, consider the smart home, where consumer IoT meets the smart grid. The interaction between these two domains elicits the need for stronger security protocols that can protect the safety-critical grid infrastructure. Smart home appliances are widely distributed, and given their intimate connection to reality, their impersonation can sometimes turn out to be very lucrative (consider ransomware for the home)—or at the same time have devastating consequences.

**The smart meter as the gateway:** With over 50 million AMI meters installed across the United States, the smart meter is one of the most pervasive smart devices along with the smartphone. In the vision of the consumer-side smart grid, every house will have a smart meter that communicates with other smart appliances in the house to make users more energy aware and energy efficient. For example:

- Meters could have knowledge of occupancy of a particular home, and turn off particular appliances to save electricity.
- The smart meter would receive real-time pricing information, and could



then relay pricing signals to the appliances.

- The smart meter could communicate demand response signals to these appliances as well. Examples include switching off air conditioning for 20 minutes or reducing the heat by 5 °F over the next hour to help ease stress on the grid.
- Smart meters communicate with certain gateway devices like Bidgely [4] to help users get more detailed bills.
- A smart appliance would receive software updates from its manufacturer.
- A smart appliance would want to send repair diagnostics to the manufacturer, to aid in quick fixing of the appliance.
- A smart meter could help coordinate charging of an electric vehicle with other local usage—and perhaps even use the battery to store power.

This communication link between the smart meters and the smart devices in a home increases the attack surface of the grid—and can extend the reach of such attacks. Effective identification and authentication on these communication links is an important step toward mitigating this increased risk.

**Scalable identity:** The embedded systems in the IoT will likely have constrained computational power and memory, and (for systems not connected to the wall) may have constrained electrical power as well. The communication channels between them may also be constrained. Thus, we need to consider the engineering impacts of the supporting cryptographic technologies. For example, in prior work [5], we found that adding cryptographic authentication to the BGP routing protocol (only a few tens of thousands of entities) kills performance.

The need for such fast and lightweight authentication is also motivated by the design requirements proposed by various smart meter certification specifications [6]. The specifications put a cap on the maximum energy used by a smart meter, thereby making it important to have fast and power efficient cryptographic schemes.

In [7], we considered some issues for the smart grid alone. Also, as we will see in Chapter 3, popular but complex cryptographic algorithms may take

unacceptably long to run on these constrained devices, forcing the need for more lightweight cryptography.

Toward this end, this thesis makes the following contributions:

- We look at scalability challenges of the existing schemes for large IoT scale populations. We specifically address the limitations of certificate revocation.
- We provide an elaborate discussion of the namespace and cryptographic complexity in the consumer-side smart grid, as an example of the IoT.
- We discuss two possible solutions to the relatively less explored problem of namespace and cryptographic complexities, which could serve as a starting point for future work in this direction.

The thesis is organized as the following

- Chapter 2 provides background required to understand the work.
- Chapter 3 illustrates the permeable nature of smart grid IoT and motivates the need for action.
- Chapter 4 provides a detailed analysis of the scalability of current certificate revocation schemes.
- Chapter 5 discusses namespace and cryptographic complexity and discusses results for the proposed solution.

# CHAPTER 2

## BACKGROUND

Ubiquitous computing is the notion of making compute omnipresent. Mark Weiser, who coined the term, envisioned a world where the computer no longer is a box with a keyboard and a mouse but an entity embedded in objects that humans interact with everyday [8]. Rechristened as the Internet of Things (IoT), and fueled by Moore’s law, this world is closer to becoming a reality. And, while we may be a few years (and privacy battles) away from achieving a Minority Report resemblance, IoT has started to take form in the smart grid, the smart home and industry.

Authentication is the process of verifying an entity’s (hereby referred to as a principal) claimed identity. The best example is a username-password scheme where the username is the claimed identity and the password is proof of identity. Another commonly used method for authentication is challenge-response; often performed when stateless authentication is needed.

While authentication answers the question: “Who are you?”; the next logical question: “What permissions do you have?” is answered by authorization. Authorization typically requires an access rule that is specified in access control lists. Although, we do not explore authorization techniques in this thesis, we would like to point out that building decentralized, lightweight mechanisms to check for these access rules is an open problem and an active area of research in ubiquitous computing.

Proof of authentication is traditionally through something you know, something you have or something you are. Successful authentication is a prerequisite to the security triad: confidentiality, integrity and availability. A principal needs to be aware of who it is sending information to (confidentiality), who it is obtaining state modifications from (integrity) and who it is

allocating a resource to (availability).

While the need for a strong authentication scheme for the Internet of Things seems very evident, work has shown that existing schemes do not scale to large populations of devices [5], [7]. Also, in instances where real-time, low-power performance is of the utmost importance, many a times designers opt for no authentication over the heavy cryptographic primitives used in computing systems today. This is evident in a 2015 Hewlett Packard security report on the state of IoT [9] that shows that 80% of the devices studied had insufficient authentication and authorization capabilities. In this chapter we discuss some of the widely used authentication schemes in distributed systems: how they work and how they scale. We also look at the changes from regular distributed systems to the IoT and the proposed authentication schemes for ubiquitous computing.

## 2.1 Authentication in Large-Scale Distributed Systems

In this section we look at two of the largest distributed systems: the web and the cellular telephony system. While we do not discuss systems like Kerberos [10] which use the Needham-Schroeder scheme [11], [12] of issuing short-lived tickets, we assure the reader that our study of these schemes shows that they are not scalable solutions to large device populations.

### 2.1.1 Authentication on the web

The Public Key Infrastructure (PKI) was established as a scalable means to build the web. Every principal on the web was to have a globally unique identity or *distinguished names*. To enable this, X.509 certificates were created that bind the unique names to public keys. In order to trust someone, there was a need to look-up their certificate in a global registry. This led to the formation of a *Certificate Authority (CA)*, which was supposed to maintain telephone book like entries of every unique name on the web. The original vision for PKI was to have a single trusted entity act as the universal CA. The CA was meant to be the universally unique authority to generate and issue all certificates and its public key was the only key that any verifying

entity needed to know. However, the logistical and economic unscalability of this technique in the practical implementation has led to multiple CAs being formed. This leads to complex trust paths and while they seem to work in a somewhat broken pattern on the web today, this will not scale to the billions of devices that will be a part of the IoT.

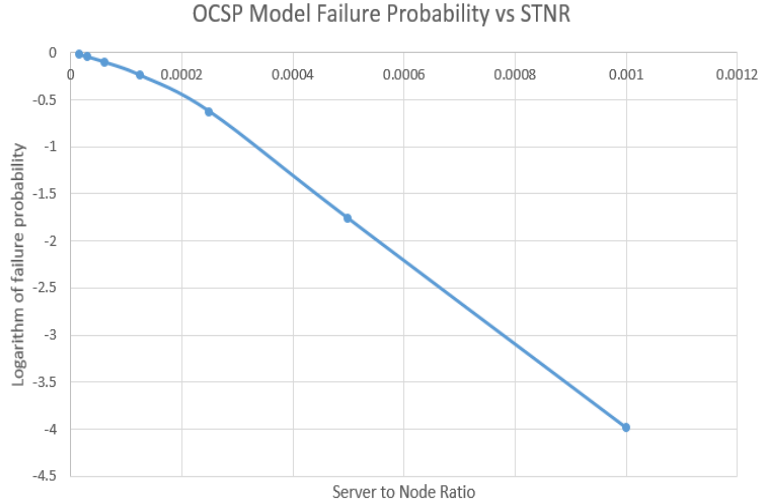


Figure 2.1: Server to node ratio vs. probability of failed validation requests for OCSP model

Another major issue with PKI is the notion of revocation. Every certificate that is issued has its validity limited by an expiry date. However, there are cases where there might be a need to *revoke* a certificate prior to the expiration date. Most revocations happen due to change in affiliation of the key holder, cessation of operation or due to private-key compromise. For trust to hold in PKI, it is important to make all nodes aware of these revocations. Many certificate revocation schemes have been proposed, however, in the Internet today, most major web browsers fail to check for revocation status. The most commonly used revocation schemes are the Certificate Revocation List (CRL), where the CA periodically publishes a list of revoked certificates to its clients, and the Online Certificate Status Protocol (OCSP), where an online check is made in real-time with a trusted entity. A recent experiment we performed showed that we would need a large number of servers that are continually online in order to have frequent and successful OCSP revocation checking. This is shown in Figure 2.1. The continually growing size

of CRLs has made them hard to read at the client side and this will only get worse with a large population the size of IoT as revocations get more frequent. OCSP, on the other hand, suffers from the fact that the trusted server is many a times unavailable due to large number of requests. When the Heartbleed vulnerability was discovered, more than 80,000 certificates needed to be revoked. Imagine finding a vulnerability like Heartbleed in a population the size of the Internet of Things and not checking for revocation. We investigate this further in Chapter 4.

Soon it was realized that certificates may work for server side authentication on the web but presented many hurdles for client side authentication. The main drawbacks of client side certificates are:

- **Unique name:** The larger the population of online principals gets, the harder it is to give them unique names. Many argue that neither is it possible to achieve global uniqueness even with the Internet size population nor is it the most relevant parameter [13].
- **User experience:** Generating and using a certificate requires effort and having the average user go through these steps do not help with usability.
- **Portability:** Since certificates are public-private key pairs, there is no way to extract the private key from say the desktop browser to use on a smartphone such that the user has the same global identity despite using different devices.
- **Privacy:** Once a user is associated to a certificate, any website can request the user's certificate. However, there may be times when the user wants to maintain a hidden identity. How a person interacts with Facebook would be totally different from how the same individual interacts with a new unknown website for the first time. If a user were to request multiple certificates to avoid using the same identity then it becomes a user experience challenge as the user is presented with a list of certificates to choose from every time a web page is visited.

Hence to avoid this process, the web uses passwords for client side authentication. Recent work at Google has aimed at making client side certifi-

cates more usable: origin bound certificates [14] are meant to work alongside passwords to provide higher security from man in the middle attacks while ensuring users retain the same user experience. Transport Layer Security (TLS) [15] provides session-oriented authentication for the web. The popularity of TLS and the returns associated with breaking it has led to it being a high-value target for attackers [16].

## 2.1.2 Authentication in GSM

The GSM telephone network shows the use of embedded symmetric keys and their use for authentication. One major drawback that is widely criticized about GSM is their principle of security by obscurity. The encryption and authentication algorithms are kept secret and not disclosed to the public (which obviously did not work ).

Every user (U) gets a Subscriber Identity Module (SIM) which is a smart card that stores a unique 128-bit symmetric key  $k_i$ . The shared version of the key is stored in a database at the authentication center (A). Whenever the user places a call the following steps occur:

$$U \rightarrow A : \text{callSetupRequest}$$

$$A \rightarrow U : \text{RAND}$$

$$U \rightarrow A : \text{MAC}_{k_i}(\text{RAND})$$

The Message Authentication Code (MAC) is calculated using the A3 algorithm which is stored on the SIM card. Another algorithm that is stored in the SIM card is the A8 algorithm. This is used to compute the session key  $k_c$ .

$$k_c : \text{MAC}_{k_i}(\text{RAND})$$

Note that this is different from the earlier MAC due to the use of a different algorithm. This thwarts any replay attacks.

Subsequently, a stream cipher A5 is used to encrypt the call traffic using the  $k_c$  that is calculated in the earlier step.

The algorithms, while meant to be secret, were leaked and this led to vulnerabilities being found and exploited. Also there is no notion of end-to-end encryption in GSM security; the connection is decrypted at the base station and re encrypted at the base station of the receiver. All communication within the system is in plain-text. Also, to get a SIM card, the user has to provide identification and fill out applications that are stored at the service provider, which makes it a user experience challenge for large systems like the IoT. These design flaws makes the protocol unscalable for the IoT population.

## 2.2 Authentication in the New Age of Computers

Section 2.1 illustrated the work done in building authentication protocols to identify principals in large-scale distributed systems. However, there are a few new challenges that are introduced by the Internet of Things that make authentication harder to perform.

- **Servers are not always reachable:** If a scheme like Kerberos were to be used, when a new refrigerator in the house wants to communicate with the AMI smart meter we need to have assurance that there is a valid ticket presented during the transaction. How does the smart meter that uses Zigbee talk to the Kerberos KDC? Does it use a gateway router? How did it come to trust the router in the first place?

If a PKI-based scheme is to be used, every certificate would need to have a valid certificate. In this case, how does the smart meter check for revocation? If it is OCSP, then how does the smart meter talk to the OCSP server? If CRL, how does the server at the CA make sure that all the nodes in the network receive up to date CRLs?

- **Secure transient association:** While security is of top value in protecting these devices, a new problem arises due to the fact that ownership of these devices is highly transient. What happens when a person



sells their washing machine on Craigslist? What if a consumer buys a new washing machine, how does that authenticate to the smart meter? Every time a physical transaction occurs, there is a change in the identity of the computer node. How does one rebind a new secure identity to the node? Having a central service that keeps track of all these transactions does not seem very scalable or usable.

In this section we explore two proposed schemes for overcoming these new obstacles.

### 2.2.1 The resurrecting duckling security policy model

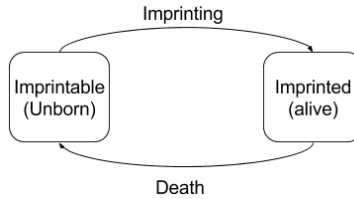


Figure 2.2: State diagram for the resurrecting duckling model

When a duckling hatches, it will recognize the first moving object, that makes a sound, as its mother. This is called imprinting. This metaphor is used to describe the resurrecting duckling security model [17], [18]. A device that is powered on for the first time will recognize as its parent the first principal that sends it a private key. In a process termed as reverse metempsychosis, a device may then be made to forget the original parent and adopt a new parent principal. The policy uses the following main references:

- **The two state principle:** The entity that is protected by the policy can only be in two states: imprinted or imprintable (shown in Figure 2.2). In the imprintable state, it is ready to receive private keys from any principal that might initiate a conversation. In the imprinted state, it acts as a slave to the principal master.
- **The imprinting principle:** When an imprinting key is sent to the entity from a principal, using a secure channel the entity is imprinted.

While it might seem easy to do, setting up a secure channel may require heavy public key cryptography. If one of the principals is powerful enough to set up a key pair and broadcast the public key then this can be simple. However, as an alternative the authors suggest using a physical contact mechanism where key transfer of the imprinting key is possible only if the two entities are in contact. This is one of the major drawbacks of the model. It is not always possible to hold two principals against each other. Also, a dedicated attacker may broadcast the imprinting key before the trusted principal can, thereby taking control of the device, even if it may be temporary.

- **The death principle:** In cases where the control of the principal needs to be transferred to another parent, the original parent may order the death of the entity thereby putting it back to an imprintable state. This can also be extended to revocation by making it a death by predefined time.
- **The assassination principle:** The device must be constructed such that it is uneconomical for an attacker to fake its death thereby putting it in an imprintable state where the attacker may send the device his private imprinting key. This requires some level of tamper resistance in the devices.

The scheme also does away with the concept of globally unique names. In the model, the duckling accepts the first principal as the mother without digging into the identity of the principal. This is promoted as *anonymous authentication* in the original paper.

## 2.2.2 The Blessings Security Model

The rate of growth of the Internet of Things with little consideration for security has put major tech companies on alert. The blessings model, by Google [19], is a security framework that tackles some of the major security

challenges.

The blessings model describes methods for identification, authentication and authorization. It promotes decentralized delegation of authority. The following definitions will help us understand the model better:

- **Identity provider:** A well-known entity that most people can agree to trust, e.g. Google, Microsoft or Facebook. This concept is derived from OpenID Connect [20] and OAuth2 [21] used in the Internet currently.
- **Principal:** Each entity in the model has a public, private key pair. For Alice,  $(P_{alice}, S_{alice})$  is considered the key pair for a principal named Alice.
- **Blessings:** Each principal has a set of hierarchical human readable strings called blessings. For example, a Samsung refrigerator owned by Alice will have the blessings: “*samsung/products/refrigerator/123*” and “*google/alice/home/refrigerator/1*”. A principal is authenticated based on its blessings. For example, all devices in Alice’s home authorize access to any principal that is “*google/alice/home/\**”.

Blessings are a chain of certificates that are bound to the principal’s public key and give the principal various identities based on the context of communication. If Alice’s refrigerator needs to talk to the maintenance shop about repairs, it will present its “*samsung/products/refrigerator/123*” blessing whereas if it wants to tell the smart meter in Alice’s house its power consumption, it will use the “*google/alice/home/refrigerator/1*” blessing.

- **Caveats:** Not every permission is created equal. Permissions grant different levels of access. The model uses caveats to capture this property [22]. For example, Alice may grant her house guest, Bob, permission to only watch the TV. This way, when Bob tries to use the gaming console, he is automatically denied access. This is stored in the blessing

certificate that Alice gives Bob. It will be of the form:

$P_{alice}$  identified as “*google/alice/*” says that  $P_{bob}$  identified as “*google/alice/guest/bob*” until time  $t$  can access “*google/alice/home/tv*”.

This notion of replacing a globally unique name with a name that is valid in the local context was first proposed as a part of the Simple Distributed Security Infrastructure (SDSI) [23] by Ron Rivest and Butler Lampson. This scheme was meant to replace the centralized PKI architecture and remove the notion of distinguished names. However, despite its success in the research community, it was not adopted into practice due to economic reasons.

Now that we understand these terms let us look at an example interaction of the bless operation. Think of Google as the root authority which creates a self signed certificate, which is referred to as the blessing root. Now, since Google is a registered identity provider, when Alice requests a certificate, it checks her identity based on its *certification practice statement* and issues her a certificate (containing her public key,  $P_{alice}$ ) signed using Google’s secret key,  $S_{google}$ . Now Alice has the blessing “*google/alice/*”. If Alice buys a new television set, she may now bless the TV by creating a certificate for the TV and signing it using her secret key  $S_{alice}$ . The TV now has a new blessing: “*google/alice/hometv*”.

Now, the thermostat in Alice’s house wants to show her a temperature notification on the TV screen, while she is watching her favorite Netflix show. The thermostat has already received a blessing, “*google/alice/thermostat*”, from Alice in the past. How does the thermostat authenticate with the TV? This mutual authentication is achieved using a process similar to the SSL/TLS setup. At the end of the transaction, both parties know each others’ blessings and if the blessings fit their access control rules then authentication succeeds. The trust is valid until one of the principals’ certificate expires. Figure 2.3 describes the process of authentication. Diffie-Hellman key exchange is used to maintain forward secrecy.

Revocation is implicitly handled in the scheme by having short-lived cer-

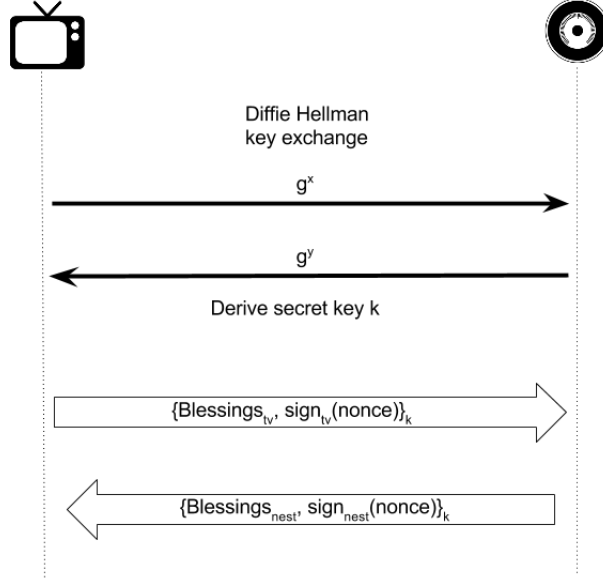


Figure 2.3: Mutual authentication in the blessings scheme

tificates. Having certificates with short validity also solves the problem of transient ownership. If a device is sold to a new owner, it will automatically request a new certificate from the new principal once the certificate expires.

The problem of online servers, however, is not completely mitigated. If the root certificate expires, the principal will need to contact the identity provider to get a new certificate. Increasing the validity of the certificate to solve this problem will impact revocation. There is also the concern of key management at constrained nodes. If a principal can have multiple blessings, a constrained node may have to manage these blessings.

Having looked at the various existing schemes and challenges associated with them, in Chapter 3, we move to our analysis of the IoT challenges that motivate the problem further.

# CHAPTER 3

## MOTIVATION

The current Internet of Computers (IoC) is composed of devices rife with security vulnerabilities. To help compensate for these endemic vulnerabilities, the current IoC depends on a “penetrate and patch” security paradigm. As holes are discovered, software is patched and eventually retired; as new attacks emerge, new signatures are pushed to anti-virus software and firewall rules are updated.

We already see IoT devices being built with the same endemic holes. In this chapter, we seek to examine the security impact of this disruption, and also to provide a foundation to examine the effectiveness to potential solution strategies.

### 3.1 IoC to IoT

Many aspects of the transition from the IoC to the IoT have the potential to disrupt the IoC’s security paradigms. Here we discuss some principal ones of concern.

**Invisibility:** As responsible users know, it can be very hard to stay on top of keeping software updated; as system administrators know, it can be very hard to convince users to be responsible. When computers stop looking like computers, we hypothesize the problem will become much worse.

**Lifetimes:** Physical devices such as appliances and light-switches will likely live much longer than laptops and personal computers; e.g., washing machines may last more than a decade, and power grid equipment may last many decades. Thus, we hypothesize that unpatched software will persist much longer. Exacerbating the problem is the fact that devices may outlive

their original ownership period—and may even outlive the company responsible for maintaining the software.

As an actual example of the hazards of device lifetime and invisibility, radiology machines with embedded computer containing old and unpatched Windows operating systems led to an infection paralyzing a hospital’s IT system [24]; as another example, air traffic systems based on even older Windows 3.1 led to the November 2016 shutdown of the Orly airport [25].

**Patchability:** We hypothesize that devices in the IoT will be harder to patch than devices in the IoC. One contributing factor is reduced network connectivity—remote devices may be hard to reach over the network; to patch the well-publicized security flaw in Jeep Cherokees, Chrysler needed to physically mail USB drives to owners. Another factor may be the fact the device software may be in unchangeable ROM or FLASH, leading to some analysts warning about the emergence of *forever-days*. A 2014 Kaspersky Labs’ study [26] showed that the CVE-2010-2568 vulnerability that was exploited by Stuxnet still remained un-patched on over 19 million computers worldwide.

**Device weakness:** We hypothesize that the security contributions of anti-virus will decrease in the IoT, as devices will be too weak to run AV (compounded by the factors above reducing the degree and effectiveness of patching).

**Consequences of compromise:** We hypothesize that the intimate connection of IoT devices to physical infrastructure will increase the damage from successful compromise. Exploding gasoline tanks [27], radio broadcasts crippling automobiles in transit [28], and blacking out northeast North America [29] are all much worse than not being able to buy things on Amazon for three days.

**Population scale:** We hypothesize that the increased size of the IoT (e.g., many tens of billions, instead of millions) will also reduce the effectiveness of penetrate and patch. The number of vulnerable systems will be larger, and the speed with which patches can propagate may in fact be longer.

## 3.2 Approaching the Problem

An absolute quantification of security, although useful in providing guarantees to user, cannot be achieved. One cannot claim that the IoT architecture is  $x\%$  secure. However, a relative quantification is possible. A relative quantification would suggest a certain improvement or a certain decline in the security of a system. While current strategies to quantify the security of a system come from pen-testing teams and security experts trying to break into a system, a new trend of modeling security has emerged in the past decade—evaluation approaches based on security metrics based and modeling the attacker (script kiddie, serious hackers and nation states) along with the cause-effect relationships between actions. In the future we hope to build models that take into account the ability of the attackers. In our ongoing work, we aim to relatively quantify the changes in security patterns and metrics from the IoC to IoT.

To build these models, we look at the past: the Common Vulnerabilities and Exposures (CVE) list maintained by MITRE. We generated a list of all reports containing specific user-provided keywords. We then parsed the list of reports to obtain the aggregate number of vulnerabilities per year relating to this keyword. We used the number of vulnerabilities reported per year to create a curve of best fit (linear or polynomial). We then used this curve of best fit to generate projections for the years 2020 and 2025.

Figure 3.1 shows the curve for server-side software; Figure 3.2 shows the curve for browser-side software; and Figure 3.3 shows the curve for device software.

## 3.3 Measuring Security Health

We want to reason about the security impact of some given number of unpatched zero-days on the IoT.

We start by considering an individual device. At any given time, it may be in one of three states:

- It may have no known vulnerabilities.
- It may have known vulnerabilities.



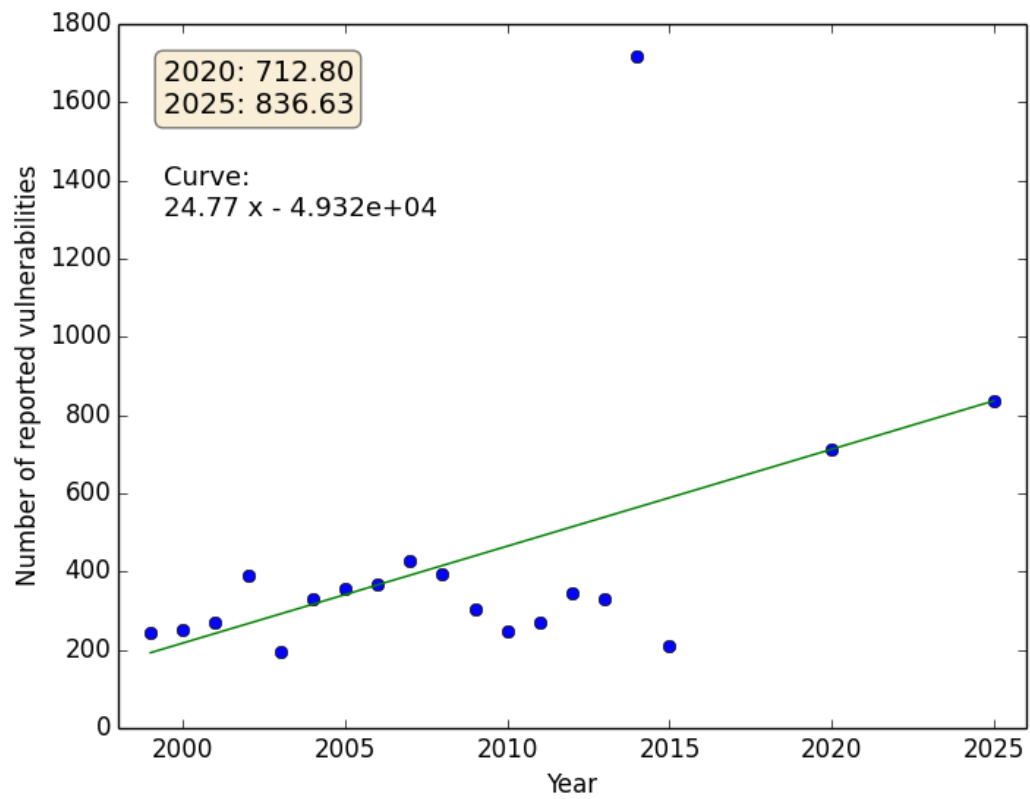


Figure 3.1: Server vulnerabilities reported in CVE through 2015—and projected to 2025

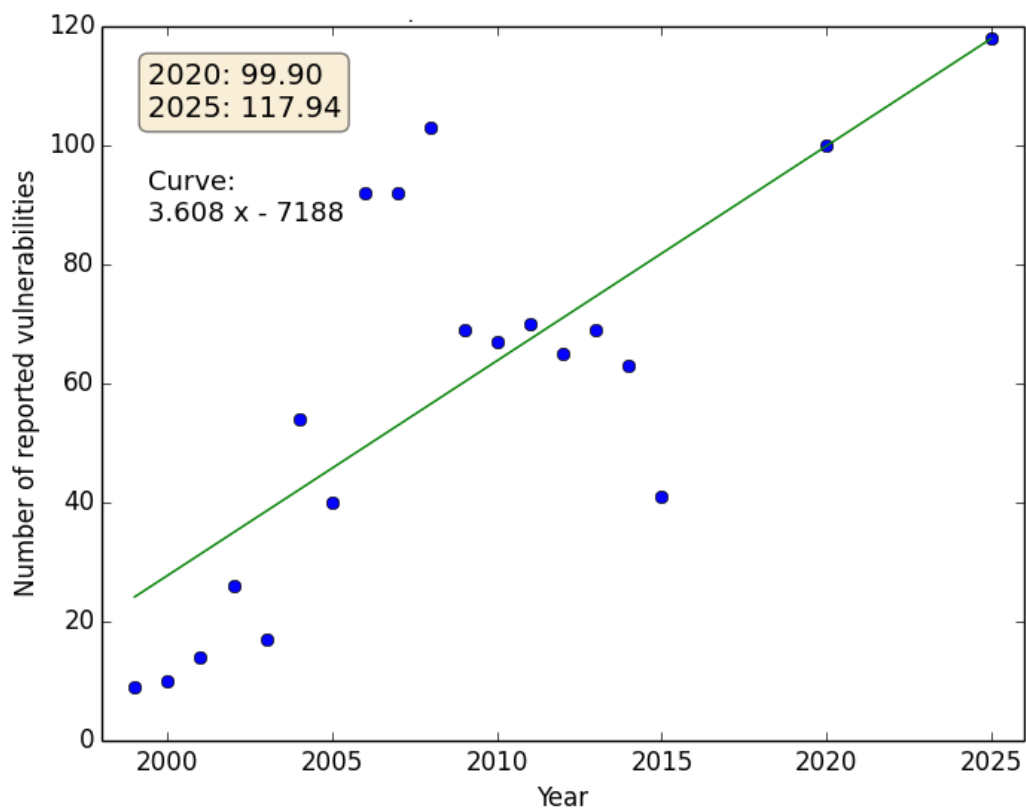


Figure 3.2: Browser vulnerabilities reported in CVE through 2015—and projected to 2025

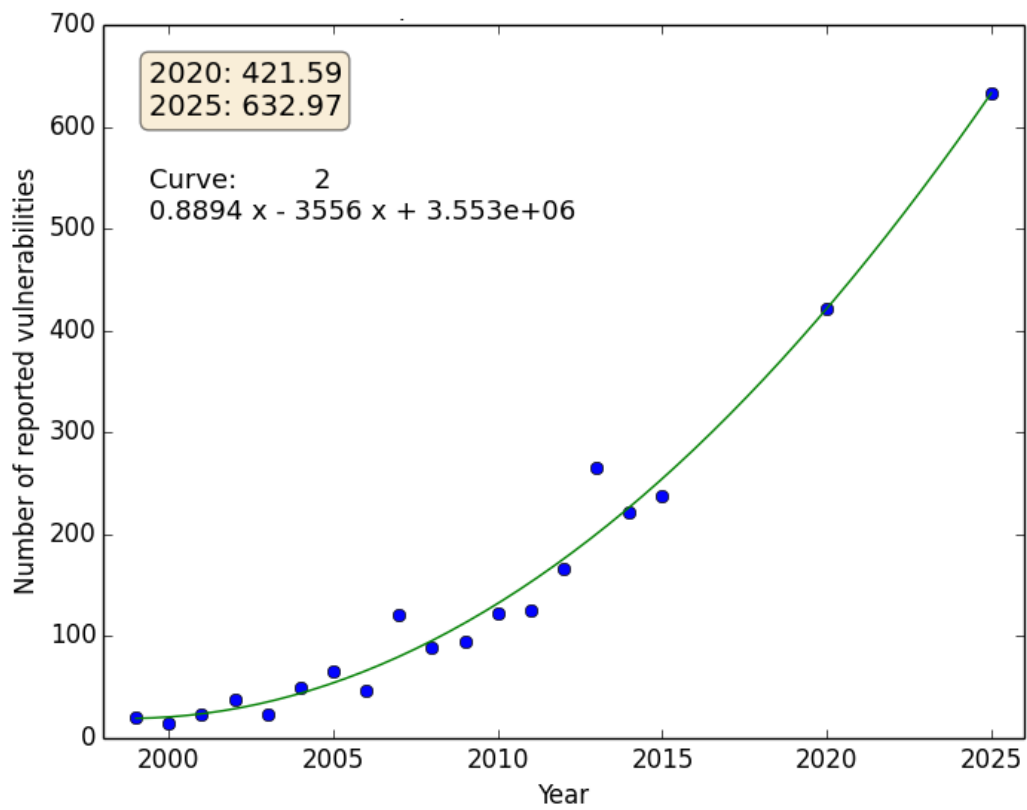


Figure 3.3: Device vulnerabilities reported in CVE through 2015—and projected to 2025

- It may be compromised.

An attack on a vulnerable device will lead to compromise, and a compromised device may then launch attacks on the devices to which it is connected. A patch issued to a vulnerable or compromised device will return it to healthy; a healthy device will become vulnerable according to a stochastic process derived from Figure 3.3. (In this initial model, for simplicity, we treat “vulnerabilities” of a device as an aggregate set instead of separate items with separate patches.)

To consider the health of the overall IoT, we might initially measure (in our model) the percentage of devices that are compromised. However, this measure alone will not capture the notion of resilience to attack: just because the devices are not currently compromised does not mean that a small focused attack cannot cause disaster. So, instead, we will measure over time:

- the percentage of devices in the system that will become compromised
- versus the percentage of devices against which an attack is launched.

A low, flat curve indicates resilience; one that grows quickly indicates trouble.

### 3.4 Modeling Health in the IoT

We build our model on the Stochastic Activity Networks (SANs) [30] formalism. SANs are an extension to Stochastic Petri Nets (SPNs). SANs use graphical primitives to provide high-level formalism which allows for detailed specification of performance models. It contains the four most important components shown in Figure 3.4.

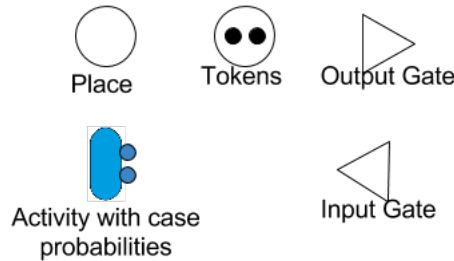


Figure 3.4: Graphical representation of SAN building blocks

- *Places* represent the state of the modeled system. Places contain *tokens* that represent the *marking* of the place. Places are like variables that contain the state or “value” of the system. Markings may either represent the number of objects, as in the number of cars that need to be washed; or markings may represent an object of a certain type, as in the priority of a task. This allows for a lot of flexibility.
- *Activities*, timed or instantaneous, are actions that take place in the modeled system over a period of time. Each timed activity has a time distribution function associated with it. When an activity fires, it causes a *transition* in the state of the system and thus changes the marking of a place.
- *Input gates* control the firing of activities and define the changes in marking that will occur after a transition takes place. Each input gate has an enabling predicate and a function. While the enabling predicate decides whether an activity must fire or not, the function defines the marking changes that must occur post activity completion.
- *Output gates* are also used to define complex completion functions. They only differ from input gates in that they can only be associated with one case if the activity has multiple cases it can choose from after it completes.

To model the IoT itself, we will consider the model shown in Figure 3.5.

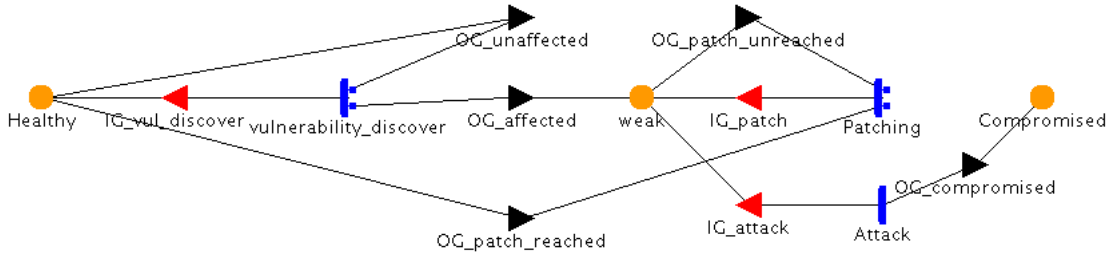


Figure 3.5: SAN for system health

- There are three places representing the three states of the model: healthy, weak or compromised.

- The markings in each place represent the number of devices in that state.
- The gates govern the transitions and contain the functions that get executed when an activity fires.
- The *vulnerability\_discover* activity is the rate at which vulnerabilities get discovered by researchers and the security community. When a vulnerability gets discovered, there is a chance that the system is affected by that vulnerability—in which case, it changes state from healthy to weak. There is also a chance that the vulnerability does not affect the system—in which case, it remains healthy.
- The *patching* activity is the rate at which patches are sent out after a vulnerability has been discovered. A patch might either reach the system or not. If the system is patched it goes from being weak to healthy; if the patch does not reach the system it remains weak. Here we define a *patchability* constant  $p$  as the probability that a patch successfully reaches a peripheral device.
- The *attack* activity defines an attack on a system before it is patched. We consider that an attack on a known vulnerability will always succeed—thus going from the weak state to the compromised state.

Figure 3.5 shows the graphical representation of the SAN model that we built. We hope to use this model as a starting point to answer questions regarding the transition from IoC to IoT.

High-level formalisms like SANs allow for ease in specification of large-scale systems. With a large model comes a large state space which may take unreasonably long to solve using analytical solutions. Discrete event simulation can be used to solve for such cases where the state spaces are too large. One drawback of discrete event simulation is the fact that if the desired measure of solving a large model is based on a “rare” event then the result may not be accurate.

In our case, we use discrete event simulation to solve the model described in this section. The fast growth of the Internet of Things population, the large number of vulnerabilities discovered every year and the even larger number of attacks on such connected environments makes it such that there are no rare events in our model. Thus, simulation is a dependable choice.

## 3.5 Initial Results

In this section we present the initial results of our analysis. We study the system with a growth in the number of IoT connected devices, an increase in vulnerabilities discovered and a raise in the number of attacks on the devices.

### 3.5.1 Parameter Choices

In our analysis we consider two variable inputs: the rate of growth of the number of devices in the Internet of Things and the rate at which vulnerabilities are discovered. The vulnerability growth profile is based on the CVE study described in Figure 3.3, which shows a steady rise in the number of vulnerabilities discovered every year. The profile for the increase in population size of the IoT is based on the intelligence study in Greenough [31]. This profile is shown in Figure 3.6. The selection of the input parameters is based on the intuition that as the number of devices in a highly interconnected environment grows, discovered vulnerabilities have a larger surface on which to spread.

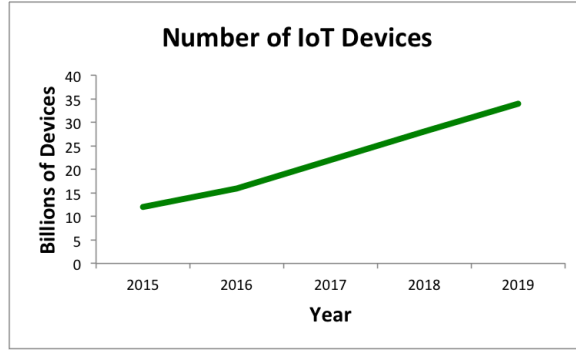


Figure 3.6: Approximation of IoT population growth based on studies in Greenough [31]

As mentioned earlier, we are interested in the relative quantification of security; hence our other parameters are conservative assumptions that help us understand the spread of attacks in a highly connected environment. We assume that every year an attack is launched against 10% of the devices in the population. This is the *attack\_rate* used in the model. We also provide a study for two cases of vulnerability patches: the first study assumes that there is a patch for every vulnerability that is discovered that year, the second

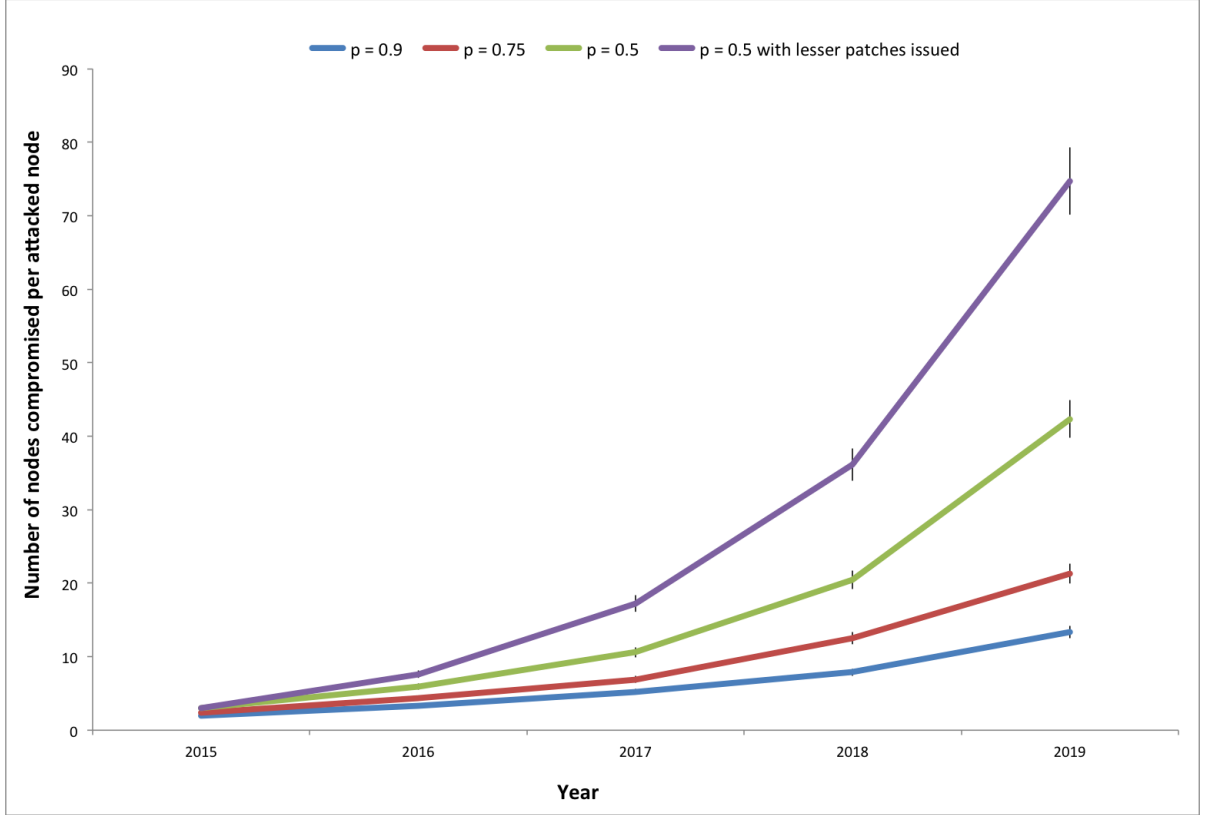


Figure 3.7: The number of nodes that get compromised per attacked node over time for different patchability constants ( $p$ ), which is the probability that a patch successfully reaches a node

study assumes that only 90% of the vulnerabilities discovered have patches. These provide the *patch\_rate* value to the model. We provide the results for both these cases in this section.

### 3.5.2 Results

Figure 3.7 describes the results we obtained. The plot aims to find out how many nodes are compromised when one node is attacked in a highly connected environment.

The first study we performed was to see the impact of change in the patchability constant. For this, we assume an ideal scenario where for every vulnerability that is discovered, a patch is issued. As we can see, even when 90% of the patches reach the nodes, there are still multiple nodes that get affected when a single node is attacked. As can be seen for the cases where



$p = 0.75$  and  $p = 0.5$ , as the patchability becomes worse, more nodes get affected by a single failed node. This shows the high amount of dependence in the IoT mesh network scenario.

The second study not only retards the patchability but also considers that only 90% of the vulnerabilities have patches issued/discovered. In this case, for every device that is attacked, potentially, more than 80 devices will be compromised.

The main goal of this study is to provide knowledge of the fact that with the Internet of Things, the probability that a patch successfully reaches every device is going to reduce drastically. As the patchability gets worse, the highly interconnected and dependent nature of the IoT would mean that a single exploitation of a vulnerability on a single node would lead to various other nodes becoming potentially compromised. While the results of this study seem intuitive, the study motivates the problem and provides a starting point for research in this direction, which is our goal. We also hope that when we use our model to study other IoT communication architectures with different levels of interdependence, we will have a better understanding of the schemes that need to be implemented in order to avoid such spreads of the attack.

Another takeaway from this initial analysis is the need for strong authentication. In the face of vulnerable device neighbors it is important for individual hosts to have a need and identity based trust relationship with its neighbors.

# CHAPTER 4

## EVALUATING THE SCALABILITY OF REVOCATION SCHEMES FOR THE INTERNET OF THINGS

Public-key cryptography was born out of a need to have a method that allowed for secure electronic key exchange in an open networked environment. The use of asymmetric-key algorithms meant that now a node in the network needed to know its own private signing key and the public verifying key of every other node in the network. The number of keys to be known was further reduced when digital certificates were introduced. Certificates allow for the node to only know its own private key and the public key of a trusted root. Public Key Infrastructure (PKI) is the term used to describe the mechanics used in establishing, maintaining, distributing and revoking digital certificates.

Digital certificates are a form of digital identification. Servers may present certificates to client computers or vice versa. In either of the cases, one entity is asserting its digital identity to another. If for some reason an aspect of the identity changes, there should be an update in the digital identification by invalidating or revoking the digital certificate associated with that entity, before re-issuing it. The Certificate Authority (CA) is entrusted with the job of issuing, validating and revoking certificates in PKI.

Every certificate that is issued, has its validity limited by an expiry date. However, there are cases where there might be a need to revoke a certificate prior to the expiration date. Most revocations happen due to change in affiliation of the key holder, cessation of operation or due to private-key compromise. For trust to hold in PKI, it is important to make all nodes aware of these revocations.

Many certificate revocation schemes have been proposed, however, in the Internet today, most major web browsers fail to check for revocation status

[32]. With the envisioned smart grid and Internet of Things, the number of nodes on the Internet are projected to reach 25 billion by the year 2020. This population is five times the size of the current Internet [33]. While, the current consensus seems to be that existing PKI techniques will scale for such large population sizes, we think there is a need to rethink certificate schemes for the high volume of low-power, low-computation capability devices that will become ubiquitous in the near future.

This chapter presents our analysis of existing revocation schemes.

## 4.1 Problem Description

The need to trust communicating entities, while important in the Internet, is enhanced in the Internet of Things due to the heavy interaction between the cyber and the physical worlds. In this section we describe the main parties involved with a certificate transaction and the two majorly used certificate revocation schemes. Both, the Certificate Revocation List (CRL) and the Online Certificate Status Protocol (OCSP) method provide a method for a node to validate if a certificate that has been presented to it has been revoked or not.

### 4.1.1 The certificate revocation model

There are three main parties involved in the certificate revocation model.

1. Certificate Authority (CA): This is a trusted party that vouches for the authenticity of the presented public key. The public key of the CA is usually known by the browser/client. The CA issues a certificate that consists of, in the most basic sense, the certificate serial number, the public key of the subject, an expiry date and the address of the server where revocation information may be found. The CA also, periodically updates a directory server with all the nodes that have been revoked since the last update.

2. Directory server: This is a non-trusted party that maintains updated certificate revocation information for the CA. It serves as a database for efficient access by clients.
3. Client: The entity that requests validation of a certificate it was presented with, by the CA.

#### 4.1.2 Certification revocation list

Certificate Revocation Lists (CRLs) [34], now the most common system, are lists of revoked certificates a client can search to determine if a presented certificate is still valid. The search process itself is not difficult, however the lists themselves become untenably long. The CA will update and publish a CRL at regular intervals. A link to the CRL is sent along with each certificate so that a client (browser) can examine the list to determine whether the certificate it is evaluating has not been recalled.

As CRLs grow, the bandwidth and memory overhead necessary to maintain CRLs across a network becomes significant. A modified CRL system, known as delta CRL, attempted to address this problem by only sending updates to CRLs across the network. Although this update significantly reduced the overhead required, transmissions still exceeded one megabyte weekly in a network of only 160,000 nodes. The overhead requirements on the multi-million node Internet of Things would be much larger, yet the memory and bandwidth available in IOT is significantly smaller; moreover previous work has also shown that using revocation lists alone fails in only one million properly-certified nodes and that revocation introduced significant security holes in client-side SSL. This scalability problem will be significantly worse in visions of the IOT.

#### 4.1.3 Online certificate status protocol

Online Certificate Status Protocol (OCSP) [35] is a newer method for checking certificate revocation status. It is a server-based validation protocol that

obviates the need for CRL transmission by requiring all users to transmit certificates to a centralized server for validation. It was designed to be a more efficient method for certificate status checks than CRLs, allowing the client computer to query a server for information about one particular certificate. The OCSP server, usually called the OCSP responder, does the necessary processing on its end and delivers a status message or assertion to the client about a single certificate. OCSP responses vary in size but the most basic kind of response, that is supported by all browsers includes: revoked, good and unknown. With CRLs, the client was required to download a potentially large file and then search through the file looking for an entry corresponding to the certificate in question. In contrast, OCSP sends a small response regarding a specific certificate, reducing both the bandwidth required as well as the amount of processing on the client. In theory OCSP's centralized model would alleviate the bandwidth and memory challenges of CRLs, however in practice OCSP introduced a new set of equally challenging problems. Users swamped the validation servers with requests, destroying real-time performance and occasionally crashing servers.

Certificate revocation checking in PKI is very important to have secure communication between two parties. Imagine an example where one of the communicating parties has its private key compromised. An attacker, with the compromised key, could launch a man in the middle attack if the certificate is not revoked or if the revoked certificate is not checked. Since the premise of PKI is based on trust, it is very important to know whom to trust at all times. Revocation means that a once trusted entity is no longer trusted and hence all nodes should update their trust. In an ideal revocation scenario, as a node is revoked, all other nodes are made aware of the revocation. However, network and memory limitations have prevented this from being a reality.

The seemingly flawed nature of the existing revocation schemes has caused major browsers to stop checking revocation status. The models described in this chapter attempt to quantify the scalability limitations of these revocation schemes. The ultimate goal is to evaluate whether the existing revocation schemes can be used for a population of devices that is five times larger than the current Internet, which seems to be the case with the envisioned Internet of Things.

## 4.2 Design Description

We studied some of the majorly proposed certificate revocation schemes [36], [37], [38] alongside CRLs and OCSP in order to find the most important aspects of a certificate revocation scheme. We found that the things that mattered most to certificate revocation were:

- **Vulnerability period:** The time interval between when a revocation is made and when the decision is made available to all the nodes in the system. A good scheme minimizes the vulnerability period.
- **Availability:** If a scheme uses an online directory server to serve revocation query requests to a set of nodes, the server should have high availability.
- **Scalability:** A good scheme must easily scale to large network sizes.
- **Overhead:** What is the bandwidth requirement? What is the response latency? Does the revocation information get stored on the device or on the gateway or on the cloud? What is the computation cost on the end device? A good scheme needs to consider reasonably the various kinds of overhead on the system and manage it.

We also asked what the differences between IoT and IoC were in order to better customize our scheme to the Internet of Things.

- **Low variance:** Unlike human users who visit a large number of web pages a day IoT devices do not actually have to talk to a wide range of servers. During the lifetime of a laptop or smartphone, the browser sets up connections with a large set of web servers. However, a constrained node generally just needs to talk to its cloud service provider. This low variance means that not a large number of public-private key pairs need to be stored on the physical device – ideally just one.

- **The MGC architecture:** A very commonly used architecture that has come to the forefront in the smart home environment is the eMbedded-Gateway-Cloud (MGC) architecture (Figure 4.1). The need for low power-consuming IoT communication protocols has allowed Bluetooth Low Energy (BLE), Zigbee and 6LoWPAN to flourish. However, these devices need a gateway system capable of translating their data into Internet packets. This gateway device is either an application on the smartphone or a physical router.

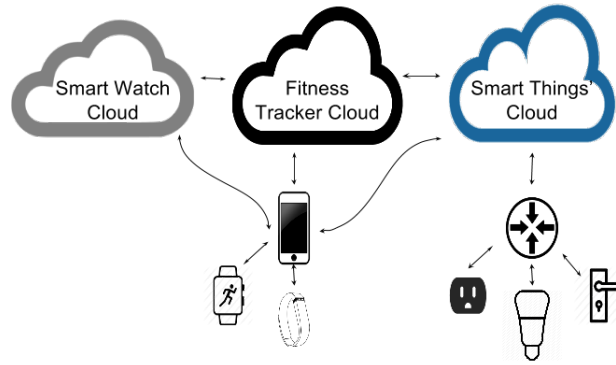


Figure 4.1: The gateway architecture

### 4.3 Modeling Approach

In this section we present the various models we built to analyze the existent revocation schemes.

We build our model on the Stochastic Activity Networks (SANs) [30] formalism. This has been described in Section 3.4.

In the following sections we evaluate the performance of both designs by developing basic SAN models to capture their behavior. We use these models to answer the following questions:

- How large do CRLs get?
- How often do the lists need to be checked?

- What is the probability of a CRL request failing?
- How many transactions fail because of specified QoS requirements?
- What percentage of transactions incorrectly succeed?
- How do the two schemes compare against each other?

#### 4.3.1 Certification revocation list models

We defined four SAN models to study the performance of Certificate Revocation Lists (CRLs). They are briefly described below.

##### CRL size model

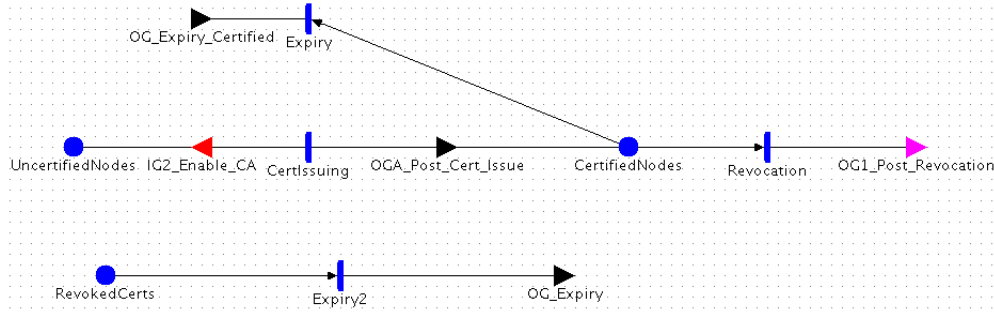


Figure 4.2: CRL size model

This model hopes to answer the question of how big the revocation lists will become with time. The larger a list, the more the memory it will require on the device. A lot of the Internet of Things devices, including smart meters in the smart grid are low-memory, low-computation devices and having a huge CRL would mean that it ultimately does not get checked.

In the model, as shown in Figure 4.2, there is a CA that issues certificates at a rate of 1000000 a year. The use of a simulation solution allows us to generate results without having to worry about the state space. The average validity of a certificate is 1 year. A certificate then expires and is no longer accepted by the browser.



An intermediate CA also revokes an average of 50 certificates a year. A revoked certificate gets added to the revocation list. It is removed from the revocation list if it expires.

### CRL latency model

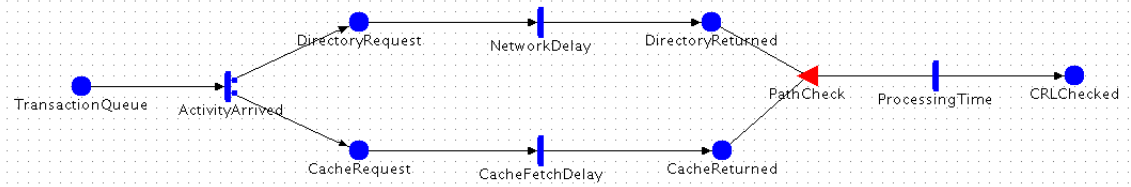


Figure 4.3: CRL latency model

In the current implementation of certificate revocation, if a CRL directory does not respond to the CRL request within a certain timeout, the browser ignores the CRL and presents the requested page. This is not a good implementation.

In this model, shown in Figure 4.3, we hope to find the probability that a CRL is returned and checked in a certain time. This can be either used to set a more effective timeout or to find the percentage of invalid transactions that pass due to a timeout.

When presented with a certificate, either the node checks its cached CRL for the revocation or it queries the directory server to get a new CRL if it has been more than 6 days (avg. time after which CA publishes a new CRL).

In the model, we have assumed a constant time for the network delay. However, in reality the network delay would depend on the size of the CRL. As the CRL size grows the network delay would increase. The processing time and cache fetching time are very small compared with the network delay.

We find transient solutions to the solution at various intervals to time in order to find the probability that the CRL has been successfully checked by that time instant.

## CRL request simulator

The CRL Request simulator SAN model (Figure 4.4) is used to simulate simultaneous CRL update requests from several nodes. In this model, a single server which can handle a fixed number of concurrent CRL requests is simulated as a *Processing\_Activity*. The server processes requests at a rate of

$$Number\_of\_Concurrent\_Requests * Processing\_Rate$$

where the *Processing\_Rate* was a fixed quantity set to a reasonable value 100 requests per hour. We think that 100 requests per hour per thread might be a reasonable processing rate because of the large CRL sizes. The set of all available nodes which simulate CRL requests to this server, are initially put into a place called *N\_nodes* at the start of the experiment. A secondary activity called the *CRL\_Update\_Activity* is defined to generate CRL requests from any of the available nodes with each node generating requests at a particular rate called the *CRL\_Update\_Rate*. A successful or an unsuccessful CRL request would include some associated network delay and this delay is incorporated into model by fixing the *CRL\_Update\_Rate* to a fixed value of 2 per hour. The specified rate value indicates that if the request was successful, then average time taken to get the CRL list is 30 mins whereas if the CRL request failed, then the node would retry on average after 30 mins. A generated CRL request is successful if the server is not already busy, i.e. the number of concurrent requests currently being served is less than a maximum allowed limit. Every successful CRL request is put into the *N\_Concurrent\_Requests* queue at the server for processing and the node which issued this request is removed from the set of available nodes. Unsuccessful requests are simply ignored and the node tries again. At the server, once a CRL request has been processed, the list is held at the corresponding nodes for an average *Hold\_Period* which is currently set to 6 days. *CRL\_Hold* activity accomplishes this task and it fires at a rate of  $Number\_of\_Served\_nodes * Hold\_Period$  and puts nodes back into the available list. We do not define a separate place for the number of served nodes but instead calculate this value using the total number of nodes, the number of available nodes and the number of concurrent requests being served.

The main purpose of this model is to study for a given Server-To-Node-Ratio (STNR), i.e. for a specified number of nodes sending CRL requests to a single server, how many requests would succeed and how many requests would fail. This is easily calculated by defining a reward variable *P\_Max\_Concurrent\_Requests* which is the probability that the server cannot accept anymore requests.

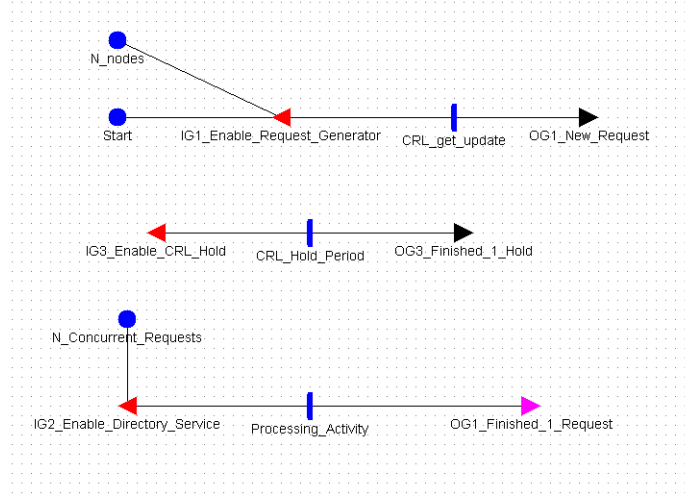


Figure 4.4: CRL request simulator

### Transaction validator

The Transaction Validator model (Figure 4.5) is used to study the outcome of a given single transaction between two initially valid nodes. In the traditional CRL architecture, nodes may not always have the most recent cached copy of CRL lists. Thus if a node involved in the transaction is revoked and the CRL list has not yet been updated at the other node, the transaction may succeed incorrectly. We use this model to obtain the probability that a given transaction would succeed incorrectly.

In our model, for the given transaction, the time at which it is executed is considered to be an event which can occur at a random time in the future. There are several events which could occur before or after the transaction is executed. The nodes involved in the transaction could get revoked after a random interval of time. The nodes involved in the transaction could also issue CRL requests repeatedly and hold the CRL lists once they are obtained.

These events are modeled as independent activities with separate rates.

*Revocation\_Activity* simulates the process of node revocation. Usually, one node involved in the transaction sends a request and the other node validates the requesting node before responding to it. The *Revocation\_Activity* fires at a fixed *Revoke\_Rate* which is the rate at which the requesting node could get revoked. To study the worst-case scenario behavior, we took a pessimistic approach and set a reasonably high average revocation time of 15 days. *Transaction\_Fire* activity simulates the process of executing the transaction. A transaction is executed after it has been fired. The *CRL\_Update\_Activity* simulates the process of issuing CRL requests at the validating node. It fires at *CRL\_Update\_Rate*. Once the *CRL\_Update\_Activity* completes, the issued CRL request may succeed or fail probabilistically. The probabilities associated with the success or failure of a CRL request are obtained from the previous model. Thus for different STNR values, we can compute the probability of incorrectly succeeding transactions. A successful CRL request would enable the *CRL\_Hold* Activity which simulates the process of holding the received CRL list for a while before re-issuing a new CRL request. An unsuccessful CRL request would simply result in re-tries.

When the CRL request activity completes, the model checks whether the requesting node has been revoked. If so, it indicates that transaction would fail because the revocation has been sensed at the validating node. Subsequently, when the transaction fires, the model checks whether the requesting node has been revoked and if so, whether the revocation was sensed. If the requesting node was revoked and it was not sensed, then the transaction is assumed to have succeeded incorrectly. We define a reward variable to detect this condition. For our evaluation, we used a transient solver and evaluated the reward variable after a long period of time. We were unable to use a steady state solver directly to compute this probability because the transaction outcome states in the model are absorbing in nature i.e. they have no outward transitions.

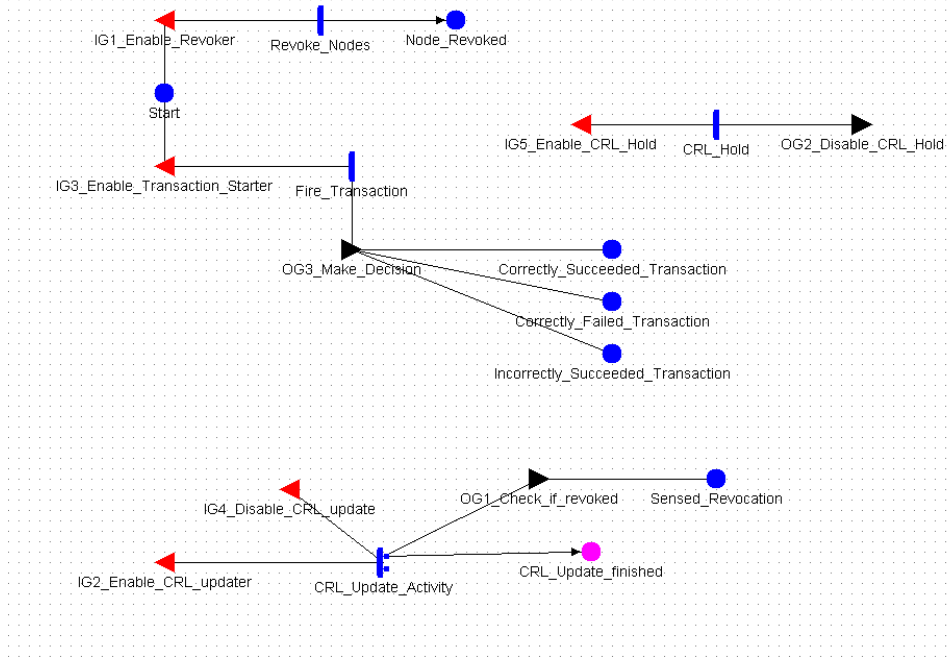


Figure 4.5: CRL transaction validator

### 4.3.2 Online certificate status protocol models

We defined three SAN models to study the performance of Online Certificate Status Protocol (OCSP). They are briefly described below.

#### OCSP latency model

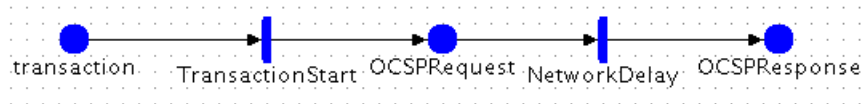


Figure 4.6: OCSP latency model

Unlike, in CRLs, everytime a certificate is presented, the node verifies its revocation status with the OCSP directory server. Also, the size of the request and response are capped at a maximum in order to allow for bandwidth restrictions. The current max is 20480 bytes. Also, the default OCSP time-out is 10 seconds. If either the size of the response is greater than the max or the request times out, the browser ignores the revocation status and presents the web page.

The model shown in Figure 4.6 assumes that the processing time for the

response is negligible. This is a valid assumption since the response only contains a status of revoked, valid or unknown.

### Transaction request simulator

The transaction request simulator SAN model (Figure 4.7) is used to simulate transaction validation requests between all participating nodes. The set of all available nodes which participate in transactions are initially put into a place called *N\_nodes* at the start of the experiment. A transaction can occur between any two nodes. An activity called *Transaction\_Generator* is defined to generate transactions between two nodes at a particular rate. The rate at which the CRL update requests are issued is  $[N\_nodes - mark()]2 * Transaction\_Generate\_Rate$ . We have currently set the *Transaction\_Generate\_Rate* to a fixed value of 12 hours to indicate the average time between two transactions a node participates in. When a transaction is generated, a request for validation is sent to the server. There is a single server which can handle a fixed number of concurrent requests. If *Number\_of\_Concurrent\_Requests* is less than the maximum allowed limit, the server can handle the request is processed it. Else the transaction simply proceeds irrespective of certificate validation. The server processes requests at a rate of  $Number\_of\_Concurrent\_Requests * Processing\_Rate$ , where the *Processing\_Rate* was a fixed quantity set to a reasonable value 1000 requests per hour. We think that 1000 requests per hour per thread might be a reasonable processing rate because there is no transmission of list involved and its a simple lookup by the server. Once the request is served, the transaction starts. The rate at which a transaction completes is defined as  $Number\_of\_Transactions\_in\_Progress * Transaction\_Complete\_Rate$ .

The *Transaction\_Complete\_Rate* is currently set to 6. This means that on average a transaction completes in 10 minutes. Once the transaction is complete, the nodes go back into the place *N\_nodes* i.e. they can start participating in other transactions.

The main purpose of this model is similar to that of the CRL request simulator model, i.e. to study for a given server to node ratio, how many requests would succeed and how many requests would fail. This is again easily calcu-

lated by defining a reward variable  $P\_Max\_Concurrent\_Requests$  which is the probability that the server cannot accept anymore requests. The reward variable is evaluated at steady state using an iterative steady state solver.

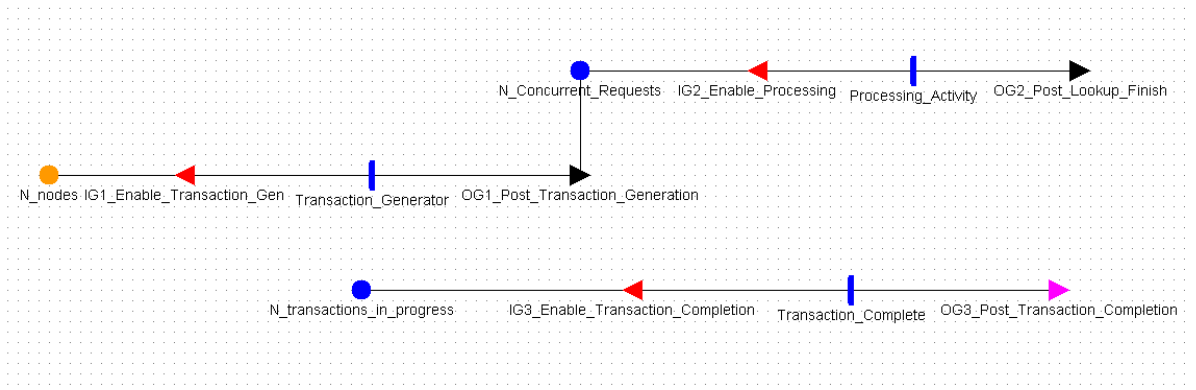


Figure 4.7: OSCP request simulator

### Transaction validator

The transaction validator model (Figure 4.8) is used to study the outcome of a given single transaction between two valid nodes when OSCP is used for certificate validation. It is similar in many ways to transaction validator model described earlier and is used to answer the same questions. In the OSCP architecture, however, the server may not always be able to serve a validation request. If a revoked node participates in a transaction and the server is busy, then the transaction proceeds anyway and succeeds incorrectly.

Similar to the earlier transaction validator model, for the given transaction, the time at which it is executed is considered to be an event which can occur at a random time in the future. The nodes involved in the transaction could get revoked after a random interval of time. This can happen before or after the transaction is executed. These two events, i.e., a transaction and a revocation are modeled as independent activities with separate rates.

*Revocation\_Activity* and *Transaction\_Fire* activities fire at the *Revoke\_Rate* and *Transaction\_Fire\_Rate* respectively. The *Transaction\_Fire* activity simulates the process of executing the transaction. There are two cases to consider after this activity completes, namely the validation server could be

busy or not. In case the server is busy with the probability calculated in the previous model, the transaction just goes ahead. Now if the initiating node has been revoked then it leads to an incorrectly successful transaction. However if the initiating node was not revoked before the transaction is executed, then the transaction succeeds correctly.

We defined a reward variable to detect the condition where a transaction incorrectly succeeds. For our evaluation, we adopted an approach similar to the transaction validator model described earlier and used a transient solver and evaluated the reward variable after a long period of time.

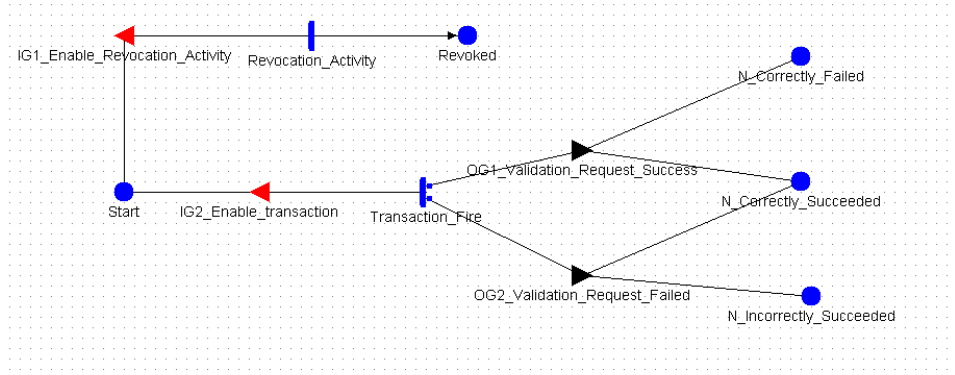


Figure 4.8: OSCP transaction validator

## 4.4 Results

### 4.4.1 Certification revocation list model

Our latency analysis on CRLs produced the cumulative distribution function shown in Figure 4.9. It is evident from the model that setting a timeout below 10 seconds, which is the average now, would mean that 20% of the transactions would timeout. As we go higher, the percentage of transactions that fail due to latency falls but the timeout then becomes too big and would affect user experience.

The study on size of CRL lists showed that in steady state 33% of the issued certificates are revoked. This is disturbing considering the memory constraint of devices. For a population size of 25 billion this means that the CRL would have about 8.25 billion revoked keys. This means a file size of



264 GB considering a key size of 256 bits. There is no way this is a scalable option.

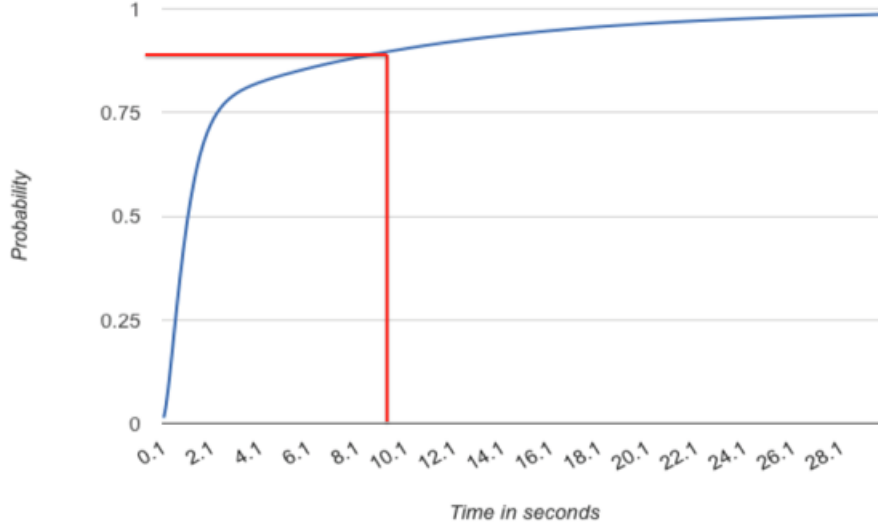


Figure 4.9: Cumulative probability distribution function of CRL checking latency

Figure 4.10 shows the plot of probability of failure vs. STNR. Here the probability of failure is the probability of a CRL request from a node to the server failing. As can be seen from the plot, as STNR decreases, i.e., in our case here the number of nodes for a single server increases, the probability of a CRL request failing increases as expected. This is so because the number of requests increase with more number of nodes, whereas there is still a single server to cater to them.

Figure 4.11 shows the plot of probability of a transaction succeeding incorrectly vs. STNR. This is an interesting result to observe for CRL model. The probability of a transaction incorrectly succeeding remains nearly the same for different STNR values we used in our analysis of the model. This shows that a revocation not being sensed and a transaction proceeding incorrectly is not influenced by the number of nodes we did our analysis on.

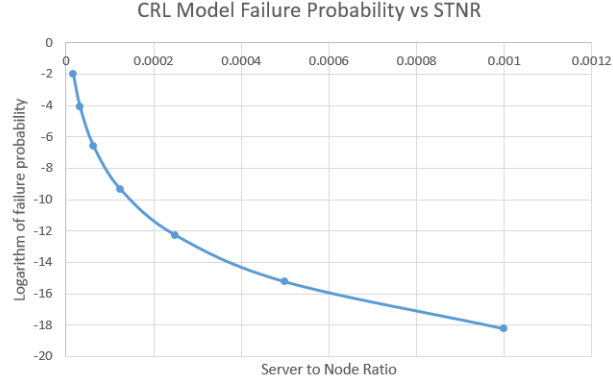


Figure 4.10: STNR vs. probability of a CRL request failing for the CRL model

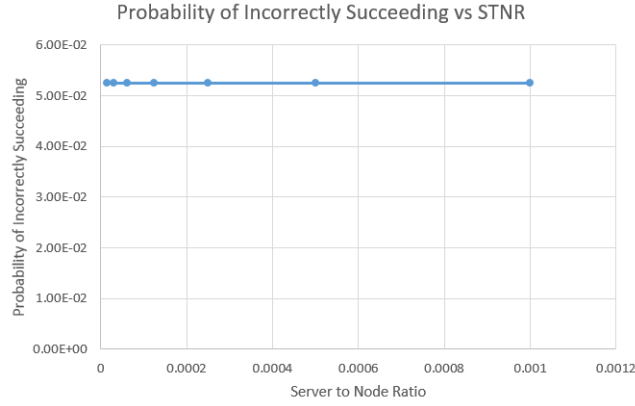


Figure 4.11: STNR vs. probability of incorrectly succeeded transactions for the CRL model

#### 4.4.2 Online certificate status protocol model

OCSP was originally created to remove the latency and bandwidth issues that were presented by CRLs. This was very evident even in the latency model that we created. Figure 4.12 shows that almost all the transactions succeed in under a second. Thus, the percentage of transactions that fail due to latency in an OCSP setting is almost zero.

Figure 4.13 shows the plot of probability of failure vs. STNR. Here the probability of failure is the probability of a validation request from a node to the server failing, due to the server being busy. As can be seen from the figure, as STNR decreases, the probability of a validation request failing increases as expected. However, the growth is much higher when compared to the CRL model which shows that OCSP is more susceptible to servers being

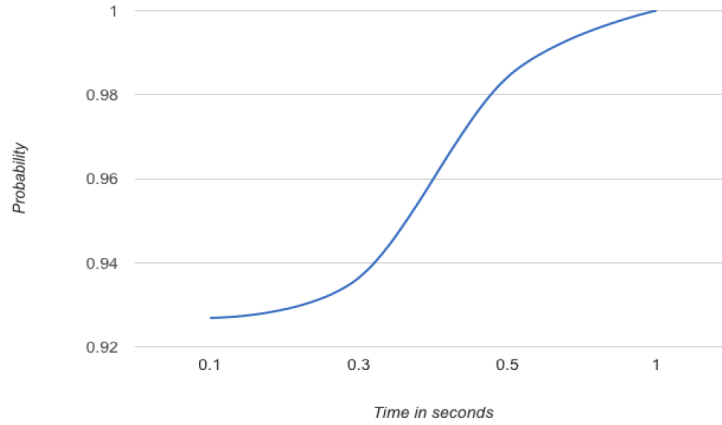


Figure 4.12: Cumulative distribution function of OCSF latency

overwhelmed.

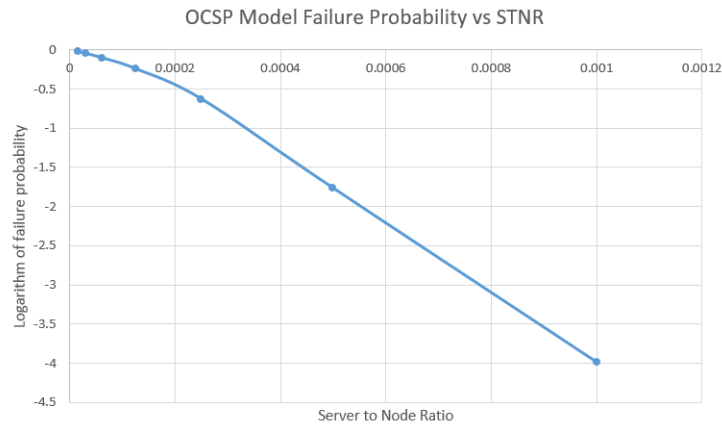


Figure 4.13: STNR vs. probability of failed validation requests for OCSF model

In Figure 4.14, we plot of probability of a transaction succeeding incorrectly as STNR is varied. A transaction succeeds incorrectly when the transaction proceeds without validation due to the server being busy, and the transaction initiating node has been revoked. The probability of a transaction incorrectly succeeding increases as STNR decreases. Higher number of nodes leads to more transactions and higher number of validation requests to the server. Due to the limited capacity of the server to handle requests an increasing number of transactions proceed without validation and hence the probability of a transaction incorrectly succeeding increases. This is in contrast to the CRL model where the probability almost remains constant thus indicating

that OCSP is more sensitive to the server's capability and the number of nodes in the system and may not be a viable, scalable solution. The requirement of secure servers makes this problem worse since that means a lower number of servers which means that in most practical cases of OCSP implementations, we would be dealing with the red region of Figure 4.14.

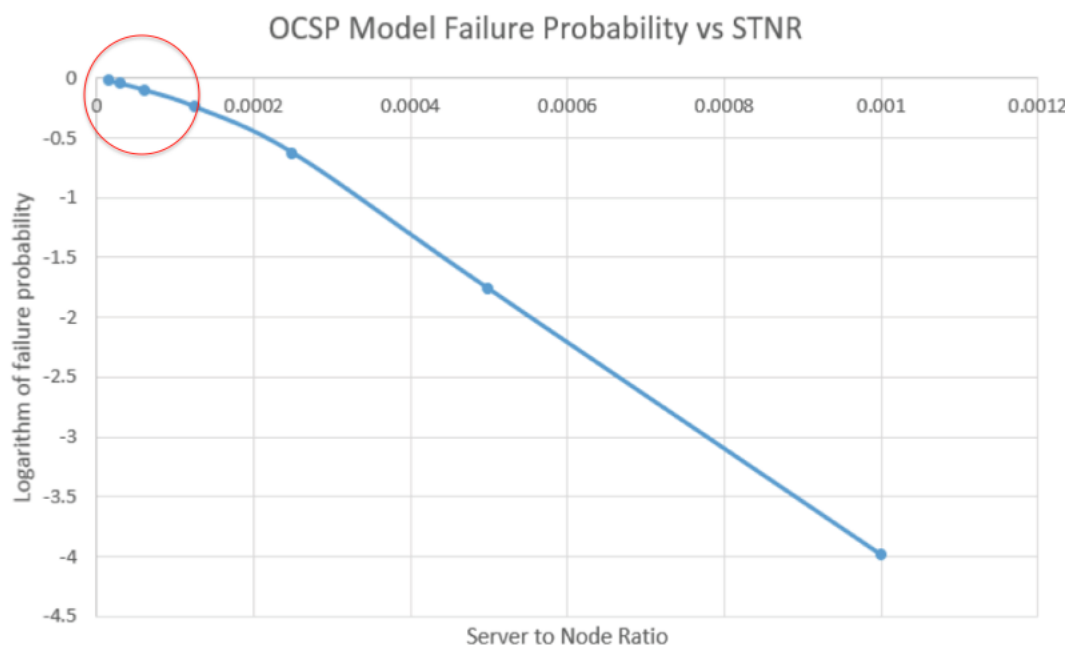


Figure 4.14: STNR vs. probability of incorrectly succeeded transactions for OCSP model

The work in this chapter illustrated that existing certificate revocation models, designed as a part of the Public Key Infrastructure (PKI) do not scale to large populations. There is a need to look at other namespace complexities that arise with certificates and to look for schemes that scale well. We look at this in Chapter 5.

# CHAPTER 5

## SCALABLE IDENTITY FOR SMART GRID IOT

For secure interaction, devices in the smart grid need to know to whom they are talking. Unfortunately, we have seen in the incipient smart grid and IoT a rush to deployment which leaves glaring holes here. Several exploits have been demonstrated including home cameras sharing private images with anyone [39]; insurance dongles in cars accepting software updates from anyone [40]; a person maliciously changing the temperature in an ex-spouse's bedroom [41]; home alarm systems allowing anyone to intercept and alter alarm messages [42].

Making things even more complicated is that a globally unique name may not suffice for the listener. What attributes do listeners need to know? Who is in a position to witness these attributes? When does it become Joe's washing machine? When will the binding change? What if Joe sells the car or moves out of the apartment? How would this communication of naming and attribute data happen? How broad or narrow are the patterns of communication that need to carry this naming and attribute data?

**Attributes:** For example, in the consumer-side smart grid, consider a basic smart appliance. If it is talking to an external party, the external party might need to know what general type of appliance it is, what its make and model are, who owns it, and the physical place it resides. The appliance may need to know who the external party is, such as: the manufacturer, a duly authorized repair person, the utility currently providing electricity to that household, or the Google calendar.

In scenarios where appliances talk to peer appliances, they need to know if this is really an appliance of a certain type, and also that the peering relationship exists (we are in the same household or perhaps even the same room together).

In scenarios where a smartphone or laptop controls devices, this controller needs to know it is talking to the right devices, and vice versa.

In electric vehicle scenarios, a car and its charging infrastructure need to authenticate each other; the charging infrastructure needs to know whose car it is, for billing, for scheduling when the charge needs to be complete, and possibly to know whether the car battery can be used as back-up for the grid.

Again, in all these situations, a simple unique global identifier (e.g. IPv6 address) does not suffice to tell the relying party what they need to know—we need names that communicate the appropriate ontology and attributes.

**Lifetimes:** In some sense, we can formalize an attribute as tuple  $(P, O, \Delta)$ : property  $P$  holds for the object  $O$  (initially, the object at the other end of the communication), for time  $\Delta$  (initially, from right now). Looking at the above discussion, we can see the need for many types of attributes in a smart home.

Some are basic identity: this  $P$  will always bind to this  $O$ . This will always be this specific tire pressure monitor, or this specific engine control unit, or this refrigerator made by GE. Someone present at the birth of this device would be in a position to assert the binding of  $P$  to  $O$ , and the binding would hold for the life of the device.

However, other attributes depend on a more dynamic “ontology of association”. How did this device come to be in this household? How did these two devices come to be in the same household? Does an appliance (or a human) change households? With what utility did the user contract? Has the tire been removed—and perhaps even sold to someone else? The likely witnesses here may even more distributed and varied—and the time interval in which the  $P, O$  binding holds may be much less than the lifetime of the device.

This latter case opens up another can of worms: How does a relying party know this witness is in a position to make this assertion?

## 5.1 Proposed Architectures

To explore scalability, we will thus consider two kinds of identity for each entity:

- **Core identity:** This would tell us that an appliance is of a particular type (e.g., washing machine of type x).
- **Association attribute:** This would tell us who or what an appliance

is associated with (e.g., washing machine in Bob’s apartment).

We will need cryptographic tools and a trust calculus so that entities can prove they have possess such identities, entities generating these assertions about other entities’ identities may themselves have supporting assertions testifying that they can do this, and relying parties will have some set of rules about how they can infer a conclusion given a set of assertions and some core axioms.

A natural approach (and one often suggested for the smart grid) is to use Public Key Infrastructure (PKI). Each entity would have matching public key and private key, and an entity can issue a digitally signed certificate asserting something about the public key of another entity. In a basic scheme, a single Certificate Authority (CA) issues certificates, any relying party who knows the CA’s public key (*trust root*) can verify a certificate, and a certificate holder proves it matches the certificate by doing something with the corresponding private key. This approach can also extend to chains of certificates (with appropriate inference rules for what these chains mean).

An interesting alternative approach is to use macaroons [22]. A macaroon consists of a *public part*—a random nonce and a set of additional data elements called *caveats*—and a *private part*—the HMAC value generated with a symmetric key on the public part. The caveats enable complex assertions like “trust this as long as it satisfies these caveats”. These caveats form the public part of the macaroon. Instead of a certificate, entities have the public part of their macaroons; instead of key pairs, entities have the private part of their macaroons. Macaroons can chain together; for example, an entity  $E_1$  with public macaroon  $M_1$  and secret  $K_1$  can use  $K_1$  to generate a macaroon  $M_2, K_2$  for entity  $E_2$ , and so on. A relying party which knows the secret key at the root of a macaroon chain can derive the private parts of all the macaroons along the chain—and thus share a secret with holder of the final macaroon.

Figure 5.1 sketches these two approaches. We now explore applying each one to the smart grid identity problem.

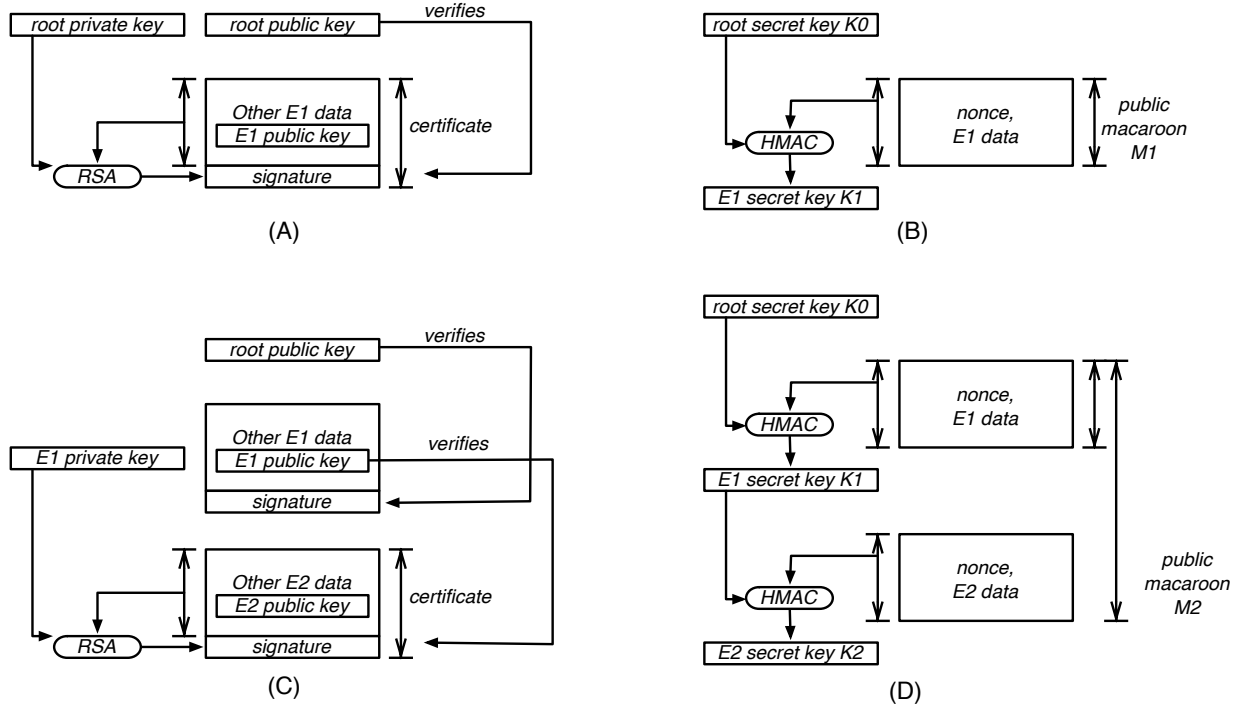


Figure 5.1: (A) With PKI, an entity exhibits a certificate to prove its identity and uses its private key to authenticate itself to any relying party knowing the trust root public key; (B) with macaroons, an entity exhibits a macaroon to prove its identity and uses this shared secret to authenticate itself to a relying party knowing the trust root secret key; (C,D) both approaches can be chained

### 5.1.1 Using PKI with attribute certificates

With PKI, we can have a trust root certify a CA at the utility and at each appliance manufacturer. The appliance manufacturer would issue an identity certificate to each appliance; the utility would issue an identity certificate to each meter. Each meter could then issue an X.509 *attribute certificate* [43], [44] to its co-located appliance, establishing the association. Figure 5.2 shows this trust flow.

**Device Registration:** When a device shows up in a house, it would present its identity certificate to the smart meter and prove knowledge of its private key. The smart meter checks the validity of the certificate, and then grants an attribute certificate which specifies that the device is associated with this specific meter.

**Revocation:** Every certificate that is issued has its validity limited by an expiry date. However, there are cases where there might be a need to revoke



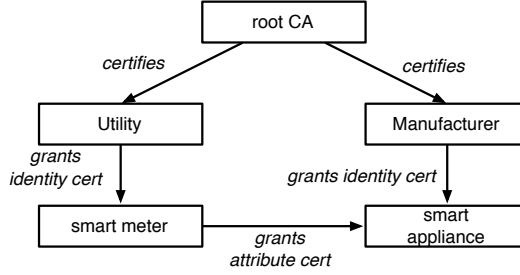


Figure 5.2: Using identity and attribute PKI for smart appliances

a certificate prior to the expiration date. Most revocations happen due to change in affiliation of the key holder, cessation of operation or private-key compromise. For trust to hold in PKI, it is important to make all nodes aware of these revocations, otherwise, relying parties may falsely conclude a binding exists when it does not. The most commonly used revocation schemes are the Certificate Revocation List (CRL), where the CA periodically publishes a list of revoked certificates to its clients, and the Online Certificate Status Protocol (OCSP), where an online check is made in real time with a trusted entity. The continually growing size of CRLs has made them hard to read at the client side and this will only get worse with a large population the size of IoT and smart grid as revocations get more frequent. OCSP, on the other hand, suffers from the fact that the trusted server is many a times unavailable due to large number of requests. Other schemes like NOVOMODO [37] and Certificate Revocation Trees (CRTs) [36] exist but they do not scale even for the Internet size population. Many certificate revocation schemes have been proposed, however, in the Internet today, most major web browsers fail to check for revocation status. When the Heartbleed vulnerability was discovered, more than 80,000 certificates needed to be revoked. Imagine finding a vulnerability like Heartbleed in a population the size of the smart grid and not checking for revocation.

**Cryptographic Considerations.** Public key cryptography is largely susceptible to brute force factoring attacks on keypair moduli. However, for moduli of at least 2048 bits (for RSA), the amount of computation required to perform brute force and crack such large keys is not in the reach of attackers. (Unfortunately, constrained devices have been shown to have crackable private keys due to being generated with predictable randomness [45]—an orthogonal problem.)

We make use of RSA, DSA [46] and the elliptic curve Ed25519 [47] in our experiments.

### 5.1.2 Using macaroons

With macaroons, we can have a trust root create an *introduction macaroon*  $M_1, K_1$  introducing a given utility's meters to a given appliance manufacturer. The utility would embed the introduction  $M_1, K_1$  with each meter. The manufacturer would use  $K_1$  to generate a *manufacturer's identity macaroon*  $M_2, K_2$  for each appliance, establishing what kind of appliance it is. When an appliance arrives in a house and presents  $M_2$  to the meter, they can set up a mutually authenticated channel using the shared secret  $K_2$ . (Using HMACs instead of digital signatures requires the manufacturer know this introduction key  $K_1$ —however, the worst it can enable the manufacturer to compromise introduction of their own appliances).

If the utility also has a root secret  $R_U$  and uses that to give each meter a *utility's identity macaroon*  $M_3, K_3$ , then the meter can then issue its own macaroons. For example, after having authenticated appliance via the introduction and manufacturer identity macaroons, the meter can issue the appliance a  $M_4, K_4$  specifying both the type and location of the appliance; the meter can also issue a pairwise introduction macaroon to pairs of appliances at the house so they can establish authenticated sessions between themselves. Figure 5.3 shows these trust flows.

The lifetimes of these macaroons would correspond to the lifetime of the bindings to which they testify.

**Shared secret secure channel:** Since there is a macaroon HMAC value that is shared between two appliances, or an appliance and the smart meter, or the smart meter or any appliance and the manufacturer themselves, we can follow the NIST guidelines [48], and use a Key Derivation Function (KDF). This KDF could be used to generate a key for the purpose of the symmetric key channel. We identify the following methods could be used for this purpose: bcrypt (72 bytes) [49], scrypt (64 bytes) [50] and argon2 (68 bytes) [51]. The standard AES encryption uses 256-bit keys (32 bytes).

The key generated by the key derivation function will have to be truncated to 256 bits to be used in AES encryption or Simon/Speck Ciphers (which

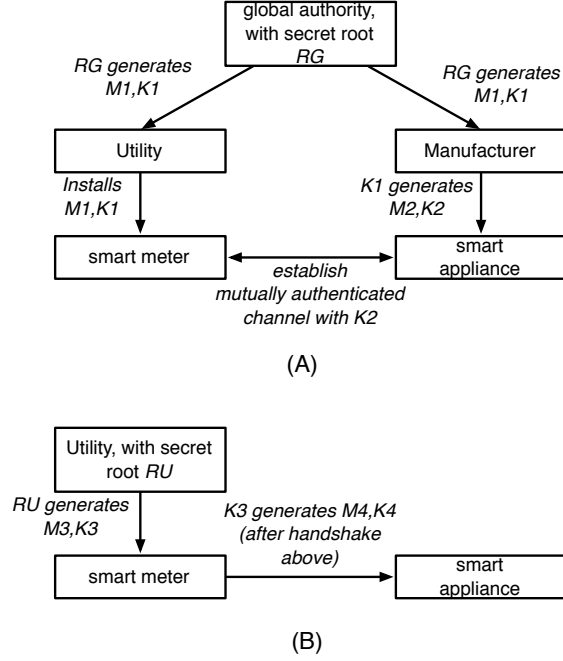


Figure 5.3: (A) To use macaroons to let a smart meter authenticate a particular type of smart appliance, a global root creates  $M1, K1$  to let a manufacturers' devices introduce themselves to a utility's meters; (B) if the utility had also given the meter its own identity macaroon beforehand, the meter can use that to to then issue a combined identity and association macaroon to the appliance

have been optimized for constrained devices [52]), which are both symmetric encryption schemes. In order to check the integrity of the messages being sent over the secure channel, we use an HMAC with MD5 and with the macaroon secret  $K$  as the initial password to generate the HMAC.

**Cryptographic considerations:** The parties generating the initial macaroons, which are namely the utility provider and a central authority, need to keep their secret keys secret, and only they can verify the correctness of the macaroon completely.

The security of the algorithm largely depends on the security of the HMAC algorithms itself. The strongest attacks against HMAC are based on the frequency of collisions, birthday attack and the timing attack could be used for improperly secured systems. The birthday attack is impractical for reasonable hash functions [53]. Even the MD5 and SHA-1 hashes, with known lack of collision resistance, do not show any vulnerabilities when used as a message authentication code [54], [55].

**Revocation:** Macaroons contain a validity caveat in them, which is implemented using epoch counters, beyond which the macaroon would no longer be valid.

The utility provider minting these macaroons, signs the macaroon with the validity caveat set to a lifespan of a few days. This macaroon is then granted to the smart meters. The smart meter could further attenuate the macaroons and transfer them to other appliances in the house.

Once the macaroon expires, the validity caveat makes the macaroon unusable. Thus, a device presenting an expired macaroon is no longer trusted.

Unlike Certificate Revocation Lists, where the utility, the smart meter and the appliances would need to check against a blacklist of invalid certificates, there is no need to maintain such state in a macaroon-based implementation. This makes it a memory and network friendly replacement.

In our next section we discuss the performance of each of the above algorithms. We also discuss the performance of attribute certificate-based algorithms and the macaroons in constrained devices.

## 5.2 Results and Discussion

In this section we discuss the performance of the smart meter in various scenarios.

The major difference between the macaroons and the PKI-based scenario is the fact that the PKI Identity and attribute certificates make use of asymmetric cryptography, where as macaroons make use of HMAC. Intuitively, we would think that an HMAC-based scheme should take much lesser time than an asymmetric scheme. We will discuss the performance further in this section.

We are concerned about testing the *createAttrCert* and *verifyAttrCert* methods of the PKI-based model and the *createMacaroon* and *verifyMacaroon* methods of the macaroon model.

The *createAttrCert* method computes a hash of the certificate contents, and then creates a signature using RSA, DSA or Elliptic Curve Ed25519. The *verifyAttrCert* algorithm verifies the signature and the hash for its correctness.

The *createMacaroon* method mints a macaroon by performing a sequence

of nested HMACs using one of the many hash functions widely available and the *verifyMacaroon* performs a similar operation and verifies the final HMAC value of the macaroon by performing the same operation.

The certificates and macaroons are minted at the initial server which has a good computational capability, and at a constrained device such as a smart meter. In the past our project colleagues have developed a smart meter research platform [56] in order to obtain realistic results for experiments pertaining to the Advanced Metering Infrastructure (AMI). The platform is an embedded system that uses a metering IC for power readings and a front-end microcontroller (TI MSP430) which runs applications for the smart meter. The platform also has a Zigbee RF module for communication between other such smart meters and to the local data collector unit. The micro-controllers and system design used to build the platform are commonly used in commercial meters that are deployed worldwide. We have modified the firmware on the smart meter research platform to incorporate the macaroons model. We also run our tests on a Raspberry Pi 2, and on a GNU/Linux server with a 3 GHz Intel Xeon CPU running at 1 GB of RAM.

We make use of the Networking and Cryptography library (NaCl) and the *PyCrypto* libraries in Python in order to perform these experiments.

We first test our PKI-based scheme by running the *createAttrCert* and *verifyAttrCert* methods on the above mentioned server and Raspberry Pi. In Table 5.1 and Table 5.2 we can see that the time taken for the tasks of creating and verifying attribute certificates. We vary the key size and the algorithm in order to check for the time taken to generate the attribute certificates. A granting authority would have to generate a keypair, then sign the certificate. For our experiments, we use a public RSA exponent value of 65537 and vary the private modulus length. Elliptic Curve Ed25519 is a high-speed signature algorithm, with fast key generation and small signatures of size 512 bits [47].

We then test our macaroons-based model, by testing our methods to create and verify the macaroons. In a macaroon-based model, the smart meter would have to add caveats and verify the macaroons it would receive. Hence, we test the macaroon-based model on the server (Table 5.3) and Raspberry Pi for different cryptographic hash algorithms (MD-5, SHA-1 and SHA-256) for computing the HMACs.

From these results, we can see that in the smart grid, it would be time

Table 5.1: Varying RSA modulus length, elliptic curve Ed25519 key size and DSA key size for PKI attribute certificates on a server

Protocol	Key length	<i>createAttrCert</i>	<i>verifyAttrCert</i>
RSA	1024 bits	40.26 ms	0.10 ms
RSA	2048 bits	253.61 ms	0.40 ms
RSA	4096 bits	1635.65 ms	1.43 ms
DSA	512 bits	19 ms	100 $\mu$ s
DSA	1024 bits	82 ms	310 $\mu$ s
Ed25519	256 bits	197 $\mu$ s	226 $\mu$ s

Table 5.2: Varying RSA modulus length, elliptic curve Ed25519 key size and DSA key size for PKI attribute certificates on a Raspberry Pi

Protocol	Key length	<i>createAttrCert</i>	<i>verifyAttrCert</i>
RSA	1024 bits	4.85 s	1.91 ms
RSA	2048 bits	24.06 s	8.33 ms
RSA	4096 bits	189.07 s	30.91 ms
DSA	512 bits	1.01 s	7.86 ms
DSA	1024 bits	1.34 s	10.36 ms
Ed25519	256 bits	25.79 ms	29.34 ms

consuming for the smart meter or any constrained device to be doing asymmetric key cryptography often. Macaroons take much less time in comparison. Moreover, the number of macaroons generated is much lesser than the number of attribute certificates, since the macaroons are not generated by a single server for every smart home appliance. If we have  $n$  smart meters, and  $m$  smart home appliances, in a macaroon-based scenario, the server would generate  $n$  macaroons, and each smart meter would generate  $m$  macaroons after attenuating them. The results are shown in the Table 5.4.

In our second set of experiments, we try to ascertain which would be the best shared secret algorithm to set up a secure channel between any two devices sharing a macaroon HMAC value. As discussed in the previous section, the methods available are using a key derivation function, to generate a key from the shared secret. These key derivation functions include the *bcrypt*, *scrypt* and the *argon2* algorithms.

The *argon2* algorithm took — 0.024 s, the *scrypt* algorithm took — 0.78 s and the *bcrypt* algorithm took — 24.41 s when tested on a raspberry pi.

While we do not explicitly do a energy consumption analysis of the two proposed schemes here, we would like to suggest that the timing analysis

Table 5.3: Varying cryptographic hash functions for an implementation of macaroons on a server

Hash Algorithm	<i>createMacaroon</i>	<i>verifyMacaroon</i>
MD5	98 $\mu$ s	79 $\mu$ s
SHA-1	100 $\mu$ s	80 $\mu$ s
SHA-256	110 $\mu$ s	85 $\mu$ s

Table 5.4: Varying cryptographic hash functions for an implementation of macaroons on constrained devices

Hash Algorithm	<i>createMacaroon</i>	<i>verifyMacaroon</i>
<b>Raspberry Pi</b>		
MD5	650 $\mu$ s	473 $\mu$ s
SHA-1	662 $\mu$ s	513 $\mu$ s
SHA-256	761 $\mu$ s	566 $\mu$ s
<b>TCIPG research platform</b>		
SHA-1	900 $\mu$ s	780 $\mu$ s
SHA-256	1.2 ms	870 $\mu$ s

in fact sheds some light on the topic. While both the schemes will have similar power draws, the total energy consumed by a scheme will in fact be determined by the time it takes to run on the smart meter processor and hence a faster scheme also relates to lower energy consumption, thereby allowing smart meters to stay within the certification requirements.

In this chapter, we explored some of the identity issues for IoT devices. We proposed two possible schemes for reliable communication in the Internet of Things. We noted that a macaroons-based scheme is expected to scale more reliably for the number of data points in the envisioned smart grid, by putting decentralization and symmetric key ciphers into practice—while still providing a lot of the flexibility of PKI-based schemes.

# CHAPTER 6

## CONCLUSION AND FUTURE WORK

In the rush to deploy more and more smart devices performing interesting tasks, we must not overlook the plumbing required to be done before deploying them. The excitement around smart devices has fuelled the consumer market to look at exciting options to make their surroundings smarter but the interaction between these smart devices often seem to have glaring security holes.

In this thesis we begin by exploring how the existence of vulnerabilities in device neighbors can lead to a bloom in attacks across the network. We then proceeded to check if existing Internet of Computer (IoC) security schemes scale to Internet of Things (IoT) size populations and analyze the Public Key Infrastructure (PKI) certificate revocation mechanism. Our findings show that revocation schemes barely work for the current size of the Internet and will definitely face challenges as the size grows.

We then proceed to look at another question that may arise: the question of unique identity in the IoT. We look at the namespace complexity for large device populations.

We finally suggest two schemes that may try to solve these problems and evaluate the schemes on a smart meter research platform to show their efficacy on low memory devices.

While in this thesis, we provide a deep exploration of the problem the solutions were just an initial glimpse at the possible solutions. We hope to, in the future, look at:

- Developing metrics to quantify absolute security of large-scale heterogeneous networks. To this end we plan to explore the space of uncertain attack graphs that model the network in the face of insufficient information and provide various attack paths that an attacker might take to the goal. As an example: How does a vulnerability in the smart washing machine lead to a potential attack at the utility company?



- Developing a large-scale simulation of the interactions between multiple IoT nodes to see how the packets transmit through the network. Use this to develop a better simulation of the zero-day blooms.
- How can the macaroons-based solution be used in other parts of the smart grid. There seems to be a potential application for using such a solution in conjunction with the MQTT protocol [57] for Industrial Control Systems (ICS).
- Look at other aspects of SCADA systems security and check scalability of existing protocols in such networks by using modeling frameworks like the one we use to study certificate revocation.

## REFERENCES

- [1] E. Fernandes, J. Jung, and A. Prakash, “Security analysis of emerging smart home applications,” in *Proceedings of the 37th IEEE Symposium on Security and Privacy*, May 2016.
- [2] A. Greenberg, “Hackers remotely kill a jeep on the highway—with me in it.” [Online]. Available: <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/>
- [3] J. Leyden, “Samsung smart fridge leaves Gmail logins open to attack.” [Online]. Available: [http://www.theregister.co.uk/2015/08/24/smart\\_fridge\\_security\\_fubar/](http://www.theregister.co.uk/2015/08/24/smart_fridge_security_fubar/)
- [4] P. Chakravarty and A. Gupta, “Impact of energy disaggregation on consumer behavior,” *Behavior, Energy and Climate Change (BECC) Conference*, 2013.
- [5] M. Zhao, S. W. Smith, and D. M. Nicol, “Aggregated path authentication for efficient BGP security,” in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, ser. CCS ’05, 2005, pp. 128–138.
- [6] I. E. Commission et al., “Electricity metering equipment (ac)—Particular requirements—Part 21: Static meters for active energy (classes 1 and 2),” 2003.
- [7] S. W. Smith, “Cryptographic scalability challenges in the smart grid (extended abstract),” in *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*, 2012, pp. 1–3.
- [8] M. Weiser, “Ubiquitous computing,” *Computer*, no. 10, pp. 71–72, 1993.
- [9] Hewlett Packard, “Internet of things research study,” 2015. [Online]. Available: <http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-4759ENW.pdf>
- [10] J. G. Steiner, B. C. Neuman, and J. I. Schiller, “Kerberos: An authentication service for open network systems,” in *USENIX Winter*, 1988, pp. 191–202.

- [11] R. M. Needham and M. D. Schroeder, “Using encryption for authentication in large networks of computers,” *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, 1978.
- [12] D. E. Denning and G. M. Sacco, “Timestamps in key distribution protocols,” *Communications of the ACM*, vol. 24, no. 8, pp. 533–536, 1981.
- [13] C. Ellison, “Improvements on conventional PKI wisdom,” in *Proceedings of the 1st Annual PKI Research Workshop*, 2002.
- [14] M. Dietz, A. Czeskis, D. Balfanz, and D. S. Wallach, “Origin-bound certificates: A fresh approach to strong client authentication for the web,” in *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 317–331.
- [15] C. Allen and T. Dierks, “The TLS protocol version 1.0,” 1999.
- [16] Y. Sheffer, R. Holz, and P. Saint-Andre, “Summarizing known attacks on transport layer security (TLS) and datagram TLS(DTLS),” 2015.
- [17] F. Stajano, “The resurrecting duckling,” in *Security Protocols*. Springer, 1999, pp. 183–194.
- [18] F. Stajano, “The resurrecting duckling: what next?” in *Security Protocols*. Springer, 2000, pp. 204–214.
- [19] M. Abadi, M. Burrows, H. Pucha, A. Sadovsky, A. Shankar, and A. Taly, “Distributed authorization with distributed grammars,” in *Programming Languages with Applications to Biology and Security*. Springer, 2015, pp. 10–26.
- [20] “Openid connect,” <http://openid.net/connect/>.
- [21] “Oauth 2.0,” <http://oauth.net/2/>.
- [22] A. Birgisson, J. G. Politz, U. Erlingsson, A. Taly, M. Vrabie, and M. Lentczner, “Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud,” in *Network and Distributed System Security Symposium*, 2014.
- [23] R. L. Rivest and B. Lampson, “SDSI: A simple distributed security infrastructure.” Crypto, 1996.
- [24] Anonymous Clinician, “Personal communication.”
- [25] P. Longeray, “Windows 3.1 is still alive, and it just killed a French airport,” November 2015. [Online]. Available: <https://news.vice.com/article/windows-31-is-still-alive-and-it-just-killed-a-french-airport>

- [26] Kaspersky Lab, “The echo of stuxnet,” August 2014. [Online]. Available: <https://securelist.com/blog/research/65367/the-echo-of-stuxnet-surprising-findings-in-the-windows-exploits-landscape/>
- [27] K. Wilhoit and S. Hilt, *The GasPot Experiment: Unexamined Perils in Using Gas-Tank-Monitoring Systems*. TrendLabs, 2015.
- [28] C. Vallance, “Car hack uses digital-radio broadcasts to seize control,” July 2015. [Online]. Available: <http://www.bbc.com/news/technology-33622298>
- [29] K. Poulsen, “Tracking the blackout bug,” April 2004. [Online]. Available: <http://www.securityfocus.com/news/8412>
- [30] W. H. Sanders and J. F. Meyer, “Stochastic activity networks: Formal definitions and concepts,” in *Lectures on Formal Methods and Performance Analysis*. Springer, 2001, pp. 315–343.
- [31] J. Greenough, “The Internet of everything: 2015,” *Business Insider Intelligence*, April 2015.
- [32] “Revocation checking and Chrome’s CRL,” <https://www.imperialviolet.org/2012/02/05/crlsets.html>.
- [33] “The size of the World Wide Web,” <http://www.worldwidewebsite.com/>.
- [34] D. Cooper, “Internet X.509 public key infrastructure certificate and certificate revocation list profile,” 2008.
- [35] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, “RFC 2560, X. 509 Internet public key infrastructure online certificate status protocol-OCSP,” *Internet Engineering Task Force*, 1999.
- [36] P. C. Kocher, “On certificate revocation and validation,” in *Financial Cryptography*, 1998, pp. 172–177.
- [37] S. Micali, “Scalable certificate validation and simplified PKI management,” in *1st Annual PKI Research Workshop*, 2002.
- [38] R. L. Rivest, “Can we eliminate certificate revocation lists?” in *Financial Cryptography*. Springer, 1998, pp. 178–183.
- [39] M. Smith, “Peeping into 73,000 unsecured security cameras thanks to default passwords,” *Network World*, November 2014.

- [40] T. Fox-Brewster, “Hacker says attacks on insecure progressive insurance dongle in 2 million US cars could spawn road carnage,” January 2015. [Online]. Available: <http://www.forbes.com/sites/thomasbrewster/2015/01/15/researcher-says-progressive-insurance-dongle-totally-insecure/>
- [41] C. Neagle, “Smart home hacking is easier than you think,” *InfoWorld*, April 2015.
- [42] C. Cimpanu, “Police body cameras shipped with pre-installed Conficker virus,” November 2015. [Online]. Available: <http://news.softpedia.com/news/police-body-cameras-shipped-with-pre-installed-conficker-virus-496177.shtml>
- [43] T. Nykänen, “Attribute certificates in x. 509,” *Techn. Ber., Helsinki University of Technology*, 2000.
- [44] D. Chadwick, A. Otenko, and E. Ball, “Role-based access control with X.509 attribute certificates,” *IEEE Internet Computing*, vol. 7, no. 2, pp. 62–69, Mar 2003.
- [45] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, “Mining your Ps and Qs: Detection of widespread weak keys in network devices,” in *USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 205–220.
- [46] D. W. Kravitz, “Digital signature algorithm,” Patent US 5 231 668, July 13, 1993.
- [47] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, “High-speed high-security signatures,” *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, 2012.
- [48] L. Chen, “Recommendation for key derivation using pseudorandom functions,” *NIST Special Publication*, vol. 800, p. 108, 2008.
- [49] N. Provos and D. Mazieres, “A future-adaptable password scheme,” in *USENIX Annual Technical Conference, FREENIX Track*, 1999, pp. 81–91.
- [50] C. Percival, “Stronger key derivation via sequential memory-hard functions,” 2009. [Online]. Available: <https://hackage.haskell.org/package/scrypt>
- [51] A. Biryukov, D. Dinu, and D. Khovratovich, “Fast and tradeoff-resilient memory-hard functions for cryptocurrencies and password hashing,” *IACR Cryptology ePrint Archive*, vol. 2015, p. 430, 2015.

- [52] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “The SIMON and SPECK lightweight block ciphers,” in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 175.
- [53] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-hashing for message authentication,” Internet Requests for Comments, RFC Editor, RFC 2104, February 1997. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2104.txt>
- [54] S. Turner and L. Chen, “Updated security considerations for the MD5 message-digest and the HMAC-MD5 algorithms,” Internet Requests for Comments, RFC Editor, RFC 6151, March 2011. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc6151.txt>
- [55] M. Bellare, “New proofs for NMAC and HMAC: Security without collision resistance,” *Journal of Cryptology*, vol. 28, no. 4, pp. 844–878, 2015.
- [56] N. Edwards, *Hardware Intrusion Detection for Supply-Chain Threats to Critical Infrastructure Embedded System*. M.S. Thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2012.
- [57] D. Locke, “Message Queuing Telemetry Transport (MQTT) v3. 1 protocol specification,” *IBM DeveloperWorks Technical Library*, 2010.