LEONARDO TÓRTORA DEVIENNE FERRAZ

**Methods to improve certificate linkage and
revocation procedures in vehicular networks**

São Paulo
2019

LEONARDO TÓRTORA DEVIENNE FERRAZ

**Methods to improve certificate linkage and
revocation procedures in vehicular networks**

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para obtenção do Título de Mestre em
Ciências.

São Paulo
2019

LEONARDO TÓRTORA DEVIENNE FERRAZ

**Methods to improve certificate linkage and
revocation procedures in vehicular networks**

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para obtenção do Título de Mestre em
Ciências.

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Marcos Antonio Simplicio
Junior

São Paulo
2019

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Ferraz, Leonardo Tórtora Devienne

Methods to improve certificate linkage and revocation procedures in vehicular networks / L. T. D. Ferraz -- versão corr. -- São Paulo, 2019.
64 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.VEÍCULOS AUTÔNOMOS 2.SEGURANÇA DE REDES 3.EMIÇÃO DE CERTIFICADOS DIGITAIS 4.PRIVACIDADE RÉVOGÁVEL 5.ATAQUE DO ANIVERSÁRIO I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

ACKNOWLEDGMENTS

I would like to thank my family, Mário, Eliana and Andressa Ferraz, for always supporting me in everything that I choose to do; all my lab colleagues, for always helping me whenever I had questions; my advisor, Prof. Marcos Antonio Simplicio Junior, for his faith in me, for his patience, and for his aid whenever I needed it; CNPq, for their support; and finally, my girlfriend, Anne Victória, for always believing in me, for her prayers and for always being by my side during the whole process.

RESUMO

FERRAZ, L.T.D. **Methods to improve certificate linkage and revocation procedures in vehicular networks**. 2019. 64 f. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2019.

Espera-se que as tecnologias de comunicação veicular, também chamadas de sistemas V2X (*Vehicle-to-everything*, em inglês), se tornem comuns no futuro, proporcionando melhor eficiência e segurança no transporte. Essa implantação planejada em larga escala, no entanto, depende criticamente de abordar alguns requisitos. Por exemplo, para prevenir abusos por motoristas, mensagens trocadas entre veículos autorizados devem ser autenticadas, o que implica na necessidade de uma Infraestrutura de Chaves Públicas Veicular (VPKI, do inglês, *Vehicular Public Key Infrastructure*). Diferente de Infraestruturas de Chaves Públicas (ICPs) tradicionais, porém, é esperado também que as VPKIs preservem a privacidade dos motoristas; em particular, que nem bisbilhoteiros, nem entidades do sistema possam identificar veículos ou rastrear seus movimentos facilmente utilizando certificados não revogados. Uma solução promissora para VPKI, que lida com tais requisitos e está entre os principais candidatos para padronização nos Estados Unidos e na Europa é o Sistema de gerenciamento de credenciais de segurança (SCMS, do inglês, *Security Credential Management System*). Nessa dissertação, com o objetivo de abordar deficiências identificadas na arquitetura do SCMS, são fornecidas três contribuições principais. Primeiro, um mecanismo para melhorar a flexibilidade do processo de revogação é descrito, permitindo que certificados e a privacidade de seus proprietários sejam revogados temporariamente de maneira eficiente; essa funcionalidade é útil, por exemplo, em caso de uma falha de software ser detectada e ser necessário a liberação de uma correção. Em segundo lugar, dois ataques do aniversário contra o processo de revogação do SCMS são detalhados e posteriormente corrigidos, assim, prevenindo a degradação de segurança do sistema com o número de certificados emitidos e revogados. Por fim, é proposto um método que simplifica a arquitetura do sistema SCMS, removendo a necessidade das chamadas Autoridades de Ligação (LAS, do inglês, *Linkage Authorities*); o que não só reduz o custo de implantação do SCMS, mas também aumenta sua segurança e privacidade devido à remoção de um potencial ponto de falha/conluio.

Palavras-Chave: Veículos autônomos. Segurança de redes. Emissão de certificados digitais. Privacidade revogável. Ataque do aniversário.

ABSTRACT

FERRAZ, L.T.D. **Methods to improve certificate linkage and revocation procedures in vehicular networks**. 2019. 64 f. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2019.

Vehicular communication technologies, also called Vehicle-to-everything (V2X) systems, are expected to become common in the future, providing better efficiency and safety in transportation. This envisioned large-scale deployment, however, critically depends on addressing some requirements. For example, to prevent abuse by drivers, messages exchanged among authorized vehicles must be authenticated, which implies the need of a Vehicular Public Key Infrastructure (VPKI). Unlike traditional Public Key Infrastructures (PKIs), though, VPKIs are also expected to preserve the drivers' privacy; in particular, neither eavesdroppers or system entities should be able to easily identify or track the movements of vehicles using non-revoked certificates. One promising VPKI solution, which copes with such requirements and is among the main candidates for standardization in the United States and Europe, is Security Credential Management System (SCMS). In this thesis, aiming to address shortcomings identified in the SCMS architecture, three main contributions are provided. First, a mechanism for improving the flexibility of revocation is described, allowing certificates and their owner's privacy to be temporarily revoked in an efficient manner; this functionality is useful, for example, in case a software malfunction is detected and a patch still needs to be released. Second, two birthday attacks against SCMS's certificate revocation process are detailed and then fixed, thus preventing the system's security degradation with the number of issued and revoked certificates. Finally, a method is proposed which simplifies SCMS's system architecture, removing the need for the so-called Linkage Authorities (LAs); this not only reduces the cost for SCMS's deployment, but also improves its security and privacy due to the removal of one potential point of failure/collusion.

Keywords: Autonomous vehicles. Network security. Issuing digital certificates. Revocable privacy. Birthday attacks.

LIST OF FIGURES

Figure 1 –	Asymmetric encryption example	18
Figure 2 –	Digital signature example	19
Figure 3 –	Classic encryption and Homomorphic encryption comparison . . .	22
Figure 4 –	Overview of SCMS’s architecture	26
Figure 5 –	SCMS’s butterfly key expansion and certificate generation	29
Figure 6 –	SCMS’s key linkage tree: LA_i generates the linkage seeds (ls) and pre-linkage values (plv) employed for certificate revocation/linkage	32
Figure 7 –	Linkage tree with <i>linkage hooks</i>	33
Figure 8 –	A more secure certificate linkage tree: adding <i>security strings</i> . . .	39
Figure 9 –	Using linkage values in SCMS (top) and in the proposed <i>LA-free</i> approach (bottom)	43
Figure 10 –	Long-term protection of security strings against birthday attacks .	51

LIST OF TABLES

Table 1	–	Components of <i>security strings</i>	40
Table 2	–	Performance of the <i>LA-free</i> solution compared to the original SCMS	54

LIST OF ACRONYMS

ACPC	Activation Codes for Pseudonym Certificates
BCAM	Binary Hash Tree based Certificate Access Management
BSM	Basic Safety Message
CA	Certificate Authority
CRL	Certificate Revocation List
DLP	Discrete Algorithm Problem
ECC	Elliptic Curve Cryptography
ECIES	Elliptic Curve Integrated Encryption Scheme
HSM	Hardware Security Module
IFAL	Issue First Activate Later
KEM	Key Encapsulation Mechanism
LA	Linkage Authority
MA	Misbehavior Authority
NIST	National Institute of Standards and Technology
PCA	Pseudonym Certificate Authority
PKI	Public Key Infrastructure
PUCA	Pseudonym scheme with User-Controlled Anonymity
RA	Registration Authority
SCMS	Security Credential Management System
SEVECOM	Secure Vehicular Communication
SHA	Secure Hash Algorithm
V2I	Vehicle-to-Infrastructure
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VPKI	Vehicular Public Key Infrastructure

GENERAL NOTATION AND SYMBOLS

G	The generator point for an elliptic curve group $\mathbb{G}$
r	A random value
A	The receiver of a message
B	The sender of a message
x, y	Plaintexts
msg	A plaintext message
cip	A ciphertext message
$sign$	A digital signature
$cert$	A digital certificate
$U, \mathcal{U}$	Public signature keys (stylized $\mathcal{U}$: reserved for PCA)
u, u	Private keys associated to U and $\mathcal{U}$ (respectively)
S	Public caterpillar key for signature
s	Private caterpillar key for signature
E	Public caterpillar key for encryption
e	Private caterpillar key for encryption
$\hat{S}$	Public cocoon key for signature
$\hat{s}$	Private cocoon key for signature
$\hat{E}$	Public cocoon key for encryption
$\hat{e}$	Private cocoon key for encryption
β	Number of cocoon keys in a batch of certificates
$\mathcal{E}(msg)$	Homomorphic encryption of a plaintext message msg
$\mathcal{D}(cip)$	Homomorphic decryption of a ciphertext cip
ls	A linkage seed
lh	A <i>linkage hook</i>
lv	A linkage value
plv	A pre-linkage value
τ	Number of time periods covered by a batch of certificates
t	Index of a time period among the τ possible
σ	Number of certificates valid in each time period
c	Index of a certificate among the σ possible
v	Number of different key usages in a time period

f_1, f_2	Functions
prf_s, prf_e	Pseudorandom functions
$Enc(\mathcal{K}, str)$	Encryption of bitstring str with key $\mathcal{K}$
$Dec(\mathcal{K}, str)$	Decryption of bitstring str with key $\mathcal{K}$
$Sig(\mathcal{K}, msg)$	Signature generation of message msg with key $\mathcal{K}$
$Ver(\mathcal{K}, msg, sign)$	Verification of signature $sign$ from message msg using key $\mathcal{K}$
$Hash(str)$	Hash of bitstring str
$str_1 \parallel str_2$	Concatenation of bitstrings str_1 and str_2
ℓ	Number of LAs (typically two)
la_id	ID of a Linkage Authority (LA)
$tree_id$	Tree identifier
$depth$	Node's depth in tree
$count$	Node's index in time period and depth
2^n	Number of entries in the attacker's table
2^m	Number of values gathered by the attacker
k	Security level

CONTENTS

1	INTRODUCTION	14
1.1	Goals	16
1.2	Improvements	16
1.3	Structure	16
2	CONCEPTS	18
2.1	Asymmetric Cryptography	18
2.1.1	Asymmetric Encryption	18
2.1.2	Digital Signature	19
2.2	Public Key Infrastructure	20
2.3	Cryptographic Hash Function	20
2.4	Birthday Attack	21
2.5	Elliptic Curve Cryptography	21
2.6	Homomorphic Encryption	22
2.6.1	Paillier	23
2.7	Summary	24
3	THE SECURITY CREDENTIAL MANAGEMENT SYSTEM	25
3.1	SCMS's Structure	26
3.2	Butterfly key expansion	28
3.3	Key linkage	30
3.3.1	Processing costs	32
3.4	Summary	32
4	CONTRIBUTIONS	33

4.1	More flexible linkage and revocation procedures: <i>Linkage Hooks</i>	33
4.1.1	Possible extensions	34
4.2	Building birthday attacks against SCMS's key linkage process and preventing them using <i>Security Strings</i>	35
4.2.1	Birthday attacks on pre-linkage values	35
4.2.2	Birthday attacks on linkage seeds	37
4.2.3	Protection against birthday attacks: <i>security strings</i>	38
4.3	Linking certificates without Linkage Authorities: <i>LA-free</i>	40
4.3.1	Privacy issues involving LAs	41
4.3.2	Generating linkage values without LAs	42
4.3.3	The <i>LA-free</i> revocation process	44
4.3.4	Detection of dishonest RA by MA	45
4.3.5	Detection of dishonest RA by PCA	46
4.4	Summary	49
5	ANALYSIS	50
5.1	<i>Linkage Hooks</i> : Performance analysis	50
5.2	<i>Security Strings</i> : Security analysis	50
5.3	<i>Security Strings</i> : Performance analysis	52
5.4	<i>LA-free</i> : Performance analysis	53
5.5	<i>LA-free</i> : Cost analysis	55
5.6	Summary	56
6	RELATED WORKS	57
6.1	SEVECOM	57
6.2	PUCA	57
6.3	SECMACE	58
6.4	IFAL, BCAM e ACPC	58

6.5 Summary	59
7 CONCLUSION	60
REFERENCES	61

1 INTRODUCTION

The past decade has witnessed a surge in digital technologies embedded in physical objects, leading to what today is known as the Internet of Things (IoT). This trend has also reached the automotive industry, which has shown a growing interest in exploring interaction models such as Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) and Vehicle-to-Pedestrian (V2P), collectively referred to as Vehicle-to-Everything (V2X) communications (HARDING et al., 2014).

V2X enables several applications aimed at improving transportation safety, efficiency, and human to machine interaction (PAPADIMITRATOS et al., 2009b). For example, information classified as Basic Safety Messages (BSMs) – which include velocity, direction and brake status – can help drivers to keep a safe distance from other vehicles while maintaining a suitable speed. The vehicle’s onboard systems can evaluate such information and, if an accident appears to be imminent, the vehicle itself can provide (semi-)automatic responses to prevent it in a timely manner. These prospects are among the motivations behind a recent publication by the United States Department of Transportation (USDOT) (NHTSA, 2017) mandating that vehicles should be capable of exchanging BSMs with each other.

Albeit promising, the large scale deployment of V2X requires addressing some challenges, especially security and privacy concerns (KHODAEI; PAPADIMITRATOS, 2015). More precisely, V2X architectures are expected to: (1) ensure the legitimacy of messages exchanged between vehicles, banning users in case of misbehavior; and (2) preserve the anonymity of honest users, so their movements cannot be easily tracked by any entity. One common approach for fulfilling these requirements is to create a Vehicular Public-Key Infrastructure (VPKI) (KHODAEI; PAPADIMITRATOS, 2015; CINCILLA; HICHAM; CHARLES, 2016). In this case, the system’s security involves issuing digital certificates to vehicles, so they can sign their own messages. Such certificates can, however, be revoked if the system detects some misbehavior, such as the transmission of invalid messages (e.g., due to a malfunction or for malicious purposes). Vehicles are then expected to verify the

authenticity of received messages, acting upon them only if signed by a non-revoked peer.

Ensuring the vehicles' privacy, in turn, requires the vehicles' certificates to not include any information that identifies their owners. Otherwise, by monitoring when and where messages signed by vehicles are broadcast, an eavesdropper can track a given target. One promising approach to alleviate such issues is to rely on pseudonym certificates (PETIT et al., 2015), in which random-like strings play the role of identifiers. By signing different messages with distinct pseudonym certificates, those messages cannot be linked to the same vehicle, thus preserving the sender's privacy. Pseudonym certificates are usually short-lived (e.g., valid for one week), which contributes to their owner's privacy and facilitates revocation. The usage of traditional, long-term credentials is then reserved for situations in which a vehicle must be identified, such as proving that it is authorized to obtain new pseudonym certificates. The frequent renewal of pseudonym certificates should be avoided, though, because vehicles are not expected to constantly have access to a reliable network connection. Hence, vehicles are usually provisioned with batches of pseudonym certificates covering current and future time periods, thus enabling their operation for a long time (e.g., years) (WHYTE et al., 2013; KUMAR; PETIT; WHYTE, 2017).

Among the many pseudonym-based security solutions for V2X (see (PETIT et al., 2015) for a survey), one of the most prominent is the Security Credential Management System (SCMS) (WHYTE et al., 2013; CAMP, 2016). Indeed, this solution is currently considered a leading candidate for protecting vehicular communications in the United States (CAMP, 2016). One of the main merits of SCMS is that vehicles can obtain arbitrarily large batches of (short-lived) pseudonym certificates from a Pseudonym Certificate Authority (PCA) by means of a single, small-sized request sent to a Registration Authority (RA). This process is such that, unless RA and PCA collude, they are unable to link a pseudonym certificate to its owner, nor to learn whether two pseudonym certificates belong to the same vehicle. In case of abuse, however, the misbehaving vehicle's privacy is annulled, while its pseudonym certificates are revoked altogether by placing a small piece of information in a Certificate Revocation List (CRL). To accomplish this, SCMS requires RA and PCA to collaborate with at least two Linkage Authorities (LAs), which supply the information inserted into pseudonym certificates for enabling such an efficient revocation procedure.

1.1 Goals

The main goal of this work is to make linkage and revocation procedures in vehicular networks more efficient. More specifically, the aims are to make these procedures more flexible, to lower the VPKI's overall deployment costs, and also to prevent security degradation the more certificates are issued.

1.2 Improvements

This work presents three independent improvements to SCMS's certificate revocation and linkage approach, enhancing its security and flexibility while reducing the underlying architecture's complexity. Specifically, (1) while the original SCMS focused just on the permanent revocation of devices, a new method is described that allows for vehicles to be revoked temporarily by including a single value in the CRL. Such capability is useful, for example, when vehicles need to be temporarily suspended, or when aiding in investigations by law enforcement authorities. In addition, (2) it is shown that SCMS's revocation procedure is prone to attacks built upon the birthday paradox to degrade the VPKI's security over time. Then an alternative method is proposed that addresses this issue with minimal overhead. Finally, (3) a mechanism is described that simplifies SCMS's overall architecture, removing the need for LAs and thus, reducing its implementation costs. Namely, SCMS's design is modified in such a manner that the LAs' roles can be securely played by PCA and RA, thus avoiding the additional point of failure/collusion represented by LAs in the original SCMS.

1.3 Structure

This dissertation is organized as follows. Chapter 2 introduces some important concepts to the understanding of this work. Chapter 3 describes the SCMS protocol, focusing on its key linkage and revocation processes. Chapter 4 presents the three improvements of this work. The first one being a method for enabling the temporary revocation and linkage of certificates. Next it is showed how to build birthday attacks against SCMS's key linkage and how to fix them, which is the second improvement. In the third improvement, it is described how to build SCMS without the need of LAs. Chapter 5 analyzes the three contributions presented in this work in terms of performance, cost and security. Chapter 6 discusses some related works, giving a broad view of the state of the art on

V2X security and privacy. Finally, Chapter 7 summarizes and concludes this work with final remarks.

2 CONCEPTS

This chapter describes some important cryptographic concepts for the full understanding of the ideas presented in the following chapters of this work.

2.1 Asymmetric Cryptography

Asymmetric cryptography, also known as public-key cryptography, is a method that uses key-pairs: public keys, which can be publicly published, and private keys, which are known only to their owners. Asymmetric cryptography is used for asymmetric encryption and for digital signatures, both of which are explained in what follows.

2.1.1 Asymmetric Encryption

Asymmetric encryption is a method to encode and decode messages that uses a public key U and a private key u as parameters for an encryption method Enc and a decryption method Dec . As seen in Figure 1, a sender B that wishes to use asymmetric encryption to send a message msg to the receiver A must obtain A 's public key, U_A , and encrypt msg with it to obtain the ciphertext $cip = Enc(U_A, msg)$. Then, B sends cip to A , who, in turn, uses its own private key u_A to compute the original message $msg = Dec(u_A, cip)$. This method provides privacy because an eavesdropper that intercepts the ciphertext cip

Figure 1 – Asymmetric encryption example

while it is being transmitted would not be able to decrypt it and read the original message msg , as it does not possess A 's private key u_A . Although it is also possible to obtain data integrity and sender's authentication via digital signatures (MENEZES; OORSCHOT; VANSTONE, 1996)

2.1.2 Digital Signature

Digital signature is a scheme in which the receiver of a message can verify that the message actually came from the sender. Given a signing algorithm Sig , a verification algorithm Ver , and a public and private key-pair U and u , respectively, the sender B can send a message msg to the receiver A with signature $sign$ in the following manner. As can be seen in Figure 2, First, B computes the signature $sign$ for the message msg using the signing algorithm Sig with its private key u_B , that is, $sign = Sig(u_B, msg)$. Then B sends msg and $sign$ to A . Upon receiving them, A uses the verification algorithm Ver to verify if the signature $sign$ is valid or not. It does so by performing calculations using B 's public key U_B , the message msg and the signature $sign$, that is, $Ver(U_B, msg, sign)$, which returns either "accept" or "reject" (PASS; SHELAT, 2010). If this algorithm returns "accept", it means that the signature $sign$ is valid, and the message msg indeed came from B . Otherwise, if it returns "reject", it can mean that someone tried to forge a message passing as B , or that someone intercepted the message msg and tampered with it, for example. Basically, a digital signature provides (RIVEST; SHAMIR; ADLEMAN, 1978):

- **Message authentication**, because it is assumed that only the sender possesses the correct private key, so only he could have generated the signature.

Figure 2 – Digital signature example

Source: Made by the author.

- **Data integrity**, because if someone tampers with the message, the verification algorithm will reject the signature and the receiver can discard the message.
- **Non-repudiation**, because since the sender is the only one who has access to its private key, he is the only one who could have produced the signature and cannot deny sending the message.

2.2 Public Key Infrastructure

A Public Key Infrastructure (PKI) permits that users exchange information securely and privately by using a public and private key-pair provided by a trusted authority. It also stores and revokes digital certificates, which are electronic documents that are used to prove that a public key in particular belongs to a certain entity. In a PKI, the Certificate Authority (CA) is the one who guarantees this fact. This entity is the one who emits, stores and signs the digital certificate, and also determines its expiration date. The Registration Authority (RA), is the one who verifies the identities of the entities that are requesting digital certificates that will be stored in the CA. A certificate can also be revoked before its normal expiration date. In this case, the certificate is included in a Certificate Revocation List (CRL), so that it can be easily verified if a certificate is valid or not. This function can be executed by the CA or by another entity in the system. Finally, there is a management system interconnecting these entities (VACCA, 2004).

2.3 Cryptographic Hash Function

A hash function is a function that has at least two properties (MENEZES; OORSCHOT; VANSTONE, 1996):

- **Compression**: the hash function maps a finite arbitrary length input bitstring to a fixed output bitstring that is called hash value or simply hash.
- **Ease of computation**: given an input and a hash function, a hash value can be calculated "easily", which is a relative parameter meaning that it can be calculated in a reasonably low amount of time.

Yet, a cryptographic hash function, besides having the aforementioned properties, also is a one-way function and its security revolves around three main properties:

- **Pre-image resistance:** to essentially every pre-specified output, it is computationally unfeasible to find any input that generates that output.
- **Second pre-image resistance:** it is computationally unfeasible to find a second input that generates the same output that a specified input.
- **Collision resistance:** it is computationally unfeasible to find any two distinct inputs that generate the same output.

There are many different types of cryptographic hash functions in existence (see (MUKUNDI, 2019) for a list). The Secure Hash Algorithm (SHA) family of hash functions are among the most commonly used, because they are published as a standard by the National Institute of Standards and Technology (NIST). The ones which are still used as of 2019 are SHA-2 and SHA-3 (NIST, 2015a, 2015b). One example of these hash functions is the SHA-256 algorithm, which has a hash value of 256 bits and a 128-bit security level.

2.4 Birthday Attack

A birthday attack is an attack built upon the birthday paradox. The birthday paradox takes its name from a mathematical problem in which one wants to determine the smaller number of people necessary in a group in order to two people from that group to share a birthday with 50% probability. The answer to this question is 23 people, which is a lower number than one would answer intuitively, hence the name “paradox”. By means of a birthday attack it is possible to find a collision in a hash function in approximately $2^{\frac{n}{2}}$ tries with 50% probability, where 2^n is the total number of distinct results of the hash function. One must always take this attack in consideration when determining the security level of a hash function (KATZ; LINDELL, 2014).

2.5 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a type of asymmetric cryptography based on elliptic curves over finite fields (MILLER, 1986; KOBLITZ, 1987). Its advantage over other asymmetric cryptography schemes is that they usually require smaller keys. Analog to the Discrete Logarithm Problem (DLP), which is the basis for various asymmetric schemes (RIVEST; SHAMIR; ADLEMAN, 1978), ECC has its own DLP, which is based on the computational unfeasibility of calculating the discrete logarithm of some random element of the elliptic curve with respect to a public base point.

The elliptic curve is a curve that satisfies the equation $Y^2 = X^3 + aX + b$ over a finite field. It also includes a point at infinity which is the identity element of the abelian group. There is a finite number of elements that satisfy the curve equation including a generator point G that generates all of these elements which form the cyclic subgroup $\mathbb{G}$. The generator point G is defined by the smallest positive number that multiplied by G equals the point at infinity. As the group is cyclic, when multiplying an integer by a point in the curve, the result will also be a point that belongs in the curve. Given the result of this multiplication and the original point, the security of the elliptic curve will depend on the computational effort of finding this multiplicand, this is the elliptic curve DLP. One example of elliptic curve cryptosystem is the Elliptic Curve Integrated Encryption Scheme (ECIES) (ANSI, 2001). It can be used as part of a Key Encapsulation Mechanism (KEM), where ECIES, an asymmetric cryptosystem, can be utilized to transmit a symmetric cryptographic key.

2.6 Homomorphic Encryption

The main advantage of homomorphic encryption is that it allows to process encrypted data. When using a homomorphic encryption system, given a plaintext and a ciphertext pair, anyone can apply a function to the latter, to obtain another encrypted ciphertext that, when decrypted, the result is the same as applying a corresponding function to the plaintext (STEHLÉ; STEINFELD, 2010). Figure 3 shows a comparison between a classic encryption and a homomorphic encryption method. In a classic encryption method, if a function f_1 is applied to a pair of plaintexts x and y and the result $f_1(x, y)$

Figure 3 – Classic encryption and Homomorphic encryption comparison

Source: Made by the author.

is encrypted using a given key $\mathcal{K}$, obtaining $Enc(\mathcal{K}, f_1(x, y))$, it won't have any relation whatsoever with the encryption of x and y before the function f_1 was applied, that is $Enc(\mathcal{K}, x)$, $Enc(\mathcal{K}, y)$. However, using a homomorphic encryption, if the plaintexts x and y are encrypted and a function f_2 (usually a different function than f_1) is applied afterwards, the result $f_2(\mathcal{E}(x), \mathcal{E}(y))$ will be the same as the encryption of $f_1(x, y)$. That is, $f_2(\mathcal{E}(x), \mathcal{E}(y)) = \mathcal{E}(f_1(x, y))$, so when $f_2(\mathcal{E}(x), \mathcal{E}(y))$ is decrypted, it results in $f_1(x, y)$. Thus, with this type of encryption, given a ciphertext, it is possible for computations to be executed on it without revealing any parts of the corresponding plaintext. A fully homomorphic encryption allows for an unbounded number of homomorphic operations to be performed on ciphertexts. While in partially homomorphic encryption it is possible to perform only one type of homomorphic operation: addition or multiplication. However, if both types of schemes are compared, the fully homomorphic encryption, despite having more versatility, is usually slower and has bigger ciphertexts than a partially homomorphic encryption system (HU, 2013).

2.6.1 Paillier

One example of a partially homomorphic encryption scheme is the Paillier cryptosystem (PAILLIER, 1999). It is an additively homomorphic cryptosystem, which means that given $\mathcal{E}(x)$, $\mathcal{E}(y)$ and the associated public key, it is possible to compute $\mathcal{E}(x + y)$.

To utilize this cryptosystem, one must begin with the key generation. The description of the key generation algorithm is the following:

1. Begin by selecting two large prime numbers p and q where $\gcd(pq, (p-1)(q-1)) = 1$.
2. Next, calculate $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$
3. Given the function L , defined by $L(z) = \frac{z-1}{n}$, choose a random integer $g \in \mathbb{Z}_{n^2}^*$, such that $\gcd(L(g^\lambda \bmod n^2), n) = 1$.
4. Then, compute $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$.
5. The public key is the pair (n, g) , while the private key is the pair (λ, μ) .

To perform the homomorphic encryption ($cip = \mathcal{E}(msg)$):

1. Given the public key pair (n, g) and a plaintext message msg , where $0 \leq msg < 1$.

2. Select a random value r where $0 < r < 1$ and $r \in \mathbb{Z}_{n^2}^*$, which can be ensured if $\gcd(r, n) = 1$.
3. The ciphertext cip can be obtained by calculating $cip = g^{msg} \cdot r^n \bmod n^2$.

Now, in order to decrypt the ciphertext cip ($msg = \mathcal{D}(cip)$):

1. Given the private key pair (λ, μ) and the ciphertext message cip , where $cip \in \mathbb{Z}_{n^2}^*$.
2. The plaintext message msg can be calculated as:

$$msg = L(cip^\lambda \bmod n^2) \cdot \mu \bmod n.$$

The homomorphic addition, in turn, can be performed by multiplying the ciphertexts, which, when decrypted, will result in the sum of the plaintexts:

$$\mathcal{D}(\mathcal{E}(msg_1) \cdot \mathcal{E}(msg_2) \bmod n^2) = msg_1 + msg_2 \bmod n.$$

2.7 Summary

This chapter presented some concepts that are necessary for the understanding of the improvements contained in this work. First, it presented the basic principles of asymmetric cryptography including asymmetric encryption and digital signature. Second, it explained what is a PKI, which is necessary for understanding what it as VPKI. After that, it was showed which are the required properties of a cryptographic hash function. Next, birthday attacks were explained, which are showed in one of the contributions of this work. Following that, elliptic curve cryptography was described, which is used in some processes in the original SCMS. Finally, it was described what is homomorphic encryption and the Paillier cryptosystem, which are part of the third contribution presented in this work. The following chapter presents the description of the SCMS protocol, which is the basis for this work.

3 THE SECURITY CREDENTIAL MANAGEMENT SYSTEM

The Security Credential Management System (SCMS), originally proposed in (WHYTE et al., 2013) and later extended in (CAMP, 2016), is a proposal that deals with revocable privacy while preventing non-colluding system entities from tracking devices. These security properties are expected to hold in the so-called “honest-but-curious” threat model: even though the system’s entities follow the correct protocols when issuing and revoking pseudonym certificates, they might engage in passive attacks, trying to use the information acquired during the protocols’ execution to their advantage (e.g., to track vehicles) (KHODAEI; PAPADIMITRATOS, 2015). SCMS is one of the leading candidates for protecting V2X security in the US (WHYTE et al., 2013; CAMP, 2016), and that is the main reason why the hereby proposed improvements are built upon its certificate linkage and revocation procedures.

In SCMS, each vehicle receives two types of certificates: an enrollment certificate, which has a long expiration time (e.g., years) and identifies legitimate devices; and multiple pseudonym certificates, each having a short validity (e.g., a few days), so $\sigma \geq 1$ certificates of this type are valid simultaneously. For privacy reasons, vehicles are expected to frequently change the pseudonym certificate employed for signing messages, thus avoiding tracking by eavesdroppers. However, the value of σ should be limited by the system to avoid “sybil-like” attacks (DOUCEUR, 2002), in which one vehicle pretends to be a platoon by signing multiple messages with different pseudonyms (MOALLA et al., 2012; ALHEETI; GRUEBLER; MCDONALD-MAIER, 2015). For example, if traffic lights give a higher priority to congested roads, such a fake platoon might receive preferential treatment even when driving on lightly loaded roads.

Figure 4 – Overview of SCMS’s architecture

Source: WHYTE et al. (2013).

3.1 SCMS’s Structure

SCMS’s design is such that batches of pseudonym certificates can be efficiently distributed to vehicles, as well as revoked in case of misbehavior by their owners. The main entities that are relevant to this work are the PCA, the RA, the MA and the LAs. These entities participate in SCMS’s two main procedures: the “butterfly key expansion”, by means of which pseudonym certificates are issued to vehicles; and “key linkage”, which allows the revocation of those certificates in case of misbehavior. For completeness, SCMS’s basic architecture are hereby described (see Figure 4 for an overview of the whole architecture):

- SCMS Manager: Guarantees a fair and efficient operation, defines directives to review revocation and misbehavior requests to ensure that they are all correct.
- Certification Services: Provides information of what types of devices are certified to receive digital certificates and specifies the certification process.
- Certificate Revocation List Store (CRLS): Distributes and stores CRLs.

- Certificate Revocation List Broadcast (CRLB): Transmits the current CRL. This can be done by satellite or roadside equipment, for example.
- Device: The final entity that sends BSMS through some equipment in the vehicle, for example.
- Enrollment Certificate Authority (ECA): Issues enrollment certificates, it works like a passport to devices and can be used to request pseudonym certificates. There can be an ECA for each region, manufacturer or device type.
- Device Configuration Manager (DCM): Provides authenticated information to devices about changes in the configuration of SCMS's components. For example, a component that changes its network address or certificate, or re-transmitting policy decisions from SCMS Manager. It also certifies to the ECA that a device is eligible to receive enrollment certificates.
- Linkage Authority (LA): Generates linkage values that are used in the certificates and support an efficient revocation. There are at least two LAs (LA1 and LA2), to prevent an operator from an LA from linking certificates that belong to the same device.
- Pseudonym Certificate Authority (PCA): Issues short-term certificates known as pseudonym certificates to devices. There can be a PCA for each region, manufacturer or device type.
- Registration Authority (RA): Validates, processes and forwards pseudonym certificate requests to the PCA.
- Request Coordination (RC): Ensures that a device does not request more than one batch of certificates for a given time period. Coordinates activities between different RAs e is only necessary if a device can request certificates from multiple RAs.
- Misbehavior Authority (MA): Processes misbehavior reports in order to identify potential misconducts from devices and, if necessary, revokes and adds them to a CRL. Initiates the process of linking a certificate identifier to the corresponding enrollment certificate and adds the latter to an internal blacklist. It has three subcomponents:
 - Internal Blacklist Manager (IBLM): Sends to the RA the necessary information to update the internal blacklist.

- Global Detection (GD): Determines which devices are executing the misbehavior.
- Certificate Revocation List Generator (CRLG): Issues certificate revocation lists to the outside world and signs them.
- Location Obscurer Proxy (LOP): Hides the location of the requesting device, changing sources addresses and, thus, preventing the linking of locations and networks addresses. When it forwards information to the MA, the LOP shuffles the reports to prevent the MA from detecting the route to the origin of the report.

3.2 Butterfly key expansion

SCMS's butterfly key expansion process allows vehicles to obtain arbitrarily large batches of (short-lived) pseudonym certificates by means of a single, small-sized request. It involves the following steps, which are illustrated in Figure 5. First, the vehicle generates two pairs of *caterpillar* private/public keys, $(s, S = s \cdot G)$ and $(e, E = e \cdot G)$, for randomly picked s and e . In addition, the vehicle also instantiates two pseudorandom functions, prf_s and prf_e , which are sent to the RA together with the public caterpillar keys S and E . A pseudorandom function is a type of function, whose outputs appear random for any given input, given the function itself was chosen at random from a family of PRFs.

The RA, in turn, uses S and prf_s to generate β public *cocoon* signature keys $\hat{S}_i = S + prf_s(i) \cdot G$, where $0 \leq i < \beta$ for an arbitrary value of β . Similarly, E and prf_e are employed in the computation of β public cocoon encryption keys $\hat{E}_i = E + prf_e(i) \cdot G$. Pairs of cocoon keys $(\hat{S}_i, \hat{E}_i)$ corresponding to different vehicles are then shuffled by the RA before being individually sent to the PCA.

After receiving a pair of cocoon keys from the RA, the PCA can either create explicit certificates or engage in a implicit certification process (CERTICOM, 2013). For explicit certificates, the PCA computes the vehicle's public signature key as $U_i = \hat{S}_i + r_i \cdot G$, for a random value r_i . The resulting U_i is then inserted into a certificate $cert_i$ together with the required metadata **meta** (e.g., the corresponding validity period and linkage values, as described later in Section 3.3). Finally, the PCA digitally signs $cert_i = (U_i, \mathbf{meta})$ with its own private key u , uses the public cocoon key $\hat{E}_i$ to encrypt both the signed certificate and the value of r_i , and once again signs the result before relaying it to the RA. As a result, only the requesting vehicle can decrypt the PCA's response with the private decryption key $e + prf_e(i)$. By doing so, the vehicle learns the public signature key U_i , and then

Figure 5 – SCMS's butterfly key expansion and certificate generation

Source: Made by the author.

computes its corresponding private signature key $u_i = s + r_i + prf_s(i)$.

For implicitly certified keys, this process is slightly different: the PCA starts by computing a credential $V_i = \hat{S}_i + r_i \cdot G$ again for a random r_i , and then creates the implicit certificate $cert_i = (V_i, meta)$; the PCA then signs this certificate to obtain $sign_i = h_i \cdot r_i + u$, where $h_i = Hash(cert_i)$, and sends back to the vehicle the pair $(cert_i, sign_i)$ encrypted with $\hat{E}_i$. The vehicle, after decrypting the PCA's response, computes $h_i = Hash(cert_i)$ and sets its own private signature key as $u_i = h_i \cdot (s + prf_s(i)) + sign_i$, whereas the corresponding public signature key takes the form $U_i = u_i \cdot G$. The validity of the public key U_i can then be implicitly verified by ascertaining that $U_i = h_i \cdot V_i + \mathcal{U}$, where $\mathcal{U}$ is

the PCA’s public signature key.

Whichever the certificate model adopted, the encrypted PCA’s response is also signed using its own private signature key, aiming to prevent an “honest-but-curious” RA from engaging in a Man-in-the-Middle (MitM) attack. Namely, without this signature, a MitM attack by the RA could be performed as follows: (1) instead of $\hat{E}_i$, the RA sends to the PCA a fake cocoon encryption key $\hat{E}_i^* = z \cdot G$, for an arbitrary value of z ; (2) the RA decrypts the PCA’s response using z , learning the value of $cert_i$; (3) the RA re-encrypts the certificate with the correct $\hat{E}_i$, sending the result to the vehicle, which proceeds with the protocol as usual; and (4) whenever a vehicle presents a pseudonym-based $cert_i$ to its counterparts, so they can validate its own public signature key U_i , the RA can link that $cert_i$ to the original request, thus identifying the corresponding vehicle. As long as the vehicle verifies the PCA’s signature on the received response, however, such MitM attempt would fail because the RA would not be able to provide a valid signature for the re-encrypted certificate generated in the attack’s step 3.

This process preserves the vehicles’ privacy as long as there is no collusion between RA and PCA. After all, the shuffling of public cocoon keys performed by the RA prevents the PCA from learning whether or not a group of keys in the batch belong to the same device. Unlinkability of public keys towards the RA, in turn, is ensured because RAs never learn the value of $cert_i$ from the PCA’s encrypted response (SIMPLICIO et al., 2018c).

3.3 Key linkage

The revocation process in SCMS avoids creating large CRLs: by placing a small piece of information in a CRL, SCMS allows multiple certificates from a vehicle to be linked together. This is accomplished by including linkage values ($1v$) as part of the pseudonym certificates’ metadata, as described in what follows.

Suppose the RA needs to create a batch of pseudonym certificates covering τ time periods, with σ pseudonym certificates valid per period, so the batch size is $\beta = \tau \cdot \sigma$. In this case, the RA chooses $\ell \geq 2$ LAs, and requests from each of them β pre-linkage values $plv_i(t, c)$, where $0 \leq t < \tau$ and $0 \leq c < \sigma$.

In response to the RA’s request, LA_i picks a random 128-bit linkage seed $1s_i(0)$. Then, it iteratively computes a τ -long hash chain (LAMPORT, 1981) of the form $1s_i(t) = Hash(la_id_i \parallel 1s_i(t-1))$, where la_id_i is LA_i ’s identifier, $\parallel$ denotes concatenation, and

$1 \leq t < \tau$. Each linkage seed $\mathbf{ls}_i(t)$ obtained in this manner is then employed in the computation of σ pre-linkage values $\mathbf{plv}_i(t, c) = \text{Enc}(\mathbf{ls}_i(t), \text{la_id}_i \parallel c)$. Finally, every $\mathbf{plv}_i(t, c)$ is truncated to a suitable length, individually encrypted and authenticated¹ using a key shared between PCA and LA_i , and sent to the RA.

After receiving the LAs' responses, the RA simply includes this (encrypted) information in the pseudonym certificate request sent to the PCA, so $\mathbf{plv}_i(t, c)$ (for $1 \leq i \leq \ell$) accompanies the c -th request for a pseudonym certificate that should be valid at time period t . The PCA, after decrypting the pre-linkage values and verifying their authenticity, computes the linkage value $\mathbf{lv}(t, c)$ by XORing those pre-linkage values together. In the usual case, which consists of two LAs, the linkage value for the c -th certificate and time period t is then computed as $\mathbf{lv}(t, c) = \mathbf{plv}_1(t, c) \oplus \mathbf{plv}_2(t, c)$.

With this approach, whenever the MA notices that a pseudonym certificate was involved in some malicious event, certificates from the same owner can be revoked altogether. This requires the collaboration of the PCA, RA, and LAs, as follows. First, the PCA associates the $\mathbf{lv}$ informed by the MA to the original pseudonym certificate request from the RA. The PCA then sends this information, as well as the corresponding pre-linkage values $\mathbf{plv}_i(t, c)$, to the RA. Subsequently, the RA does the following: (1) identifies the vehicle behind the original request, placing its enrollment certificate in a blacklist to prevent it from obtaining new pseudonym certificates; and (2) asks LA_i to identify the linkage seed $\mathbf{ls}_i(0)$ from which $\mathbf{plv}_i(t, c)$ was computed. Each LA_i responds with $\mathbf{ls}_i(t_s)$, so the revocation (and, consequently, privacy loss) starts being valid at time period t_s . For example, t_s might be the current time period, or the time period when the misbehavior was first detected. The linkage seeds $\mathbf{ls}_i(t_s)$ provided by the LAs are then placed in a public CRL. This allows any entity to compute $\mathbf{lv}(t, c)$ for time periods $t \geq t_s$, identifying which certificates correspond to a CRL entry. Such mechanism ensures forward privacy: the misbehaving vehicle's certificates for *current* and *future* time periods are revoked, and messages signed with the corresponding keys can be traced back to that vehicle; messages signed in *past* time periods cannot be linked, though, preserving the user's privacy prior to the detection of the malicious activity.

¹Authentication and freshness are not explicitly mentioned in (WHYTE et al., 2013; CAMP, 2016). However, they are important properties to prevent a dishonest RA from delivering a forged or reused $\mathbf{plv}$ without contacting LA_i , thus becoming able to track vehicles, as further discussed in Section 4.3.1.

Figure 6 – SCMS’s key linkage tree: LA_i generates the linkage seeds (ls) and pre-linkage values (plv) employed for certificate revocation/linkage

Source: Made by the author.

3.3.1 Processing costs

In terms of complexity, SCMS’s revocation process is such that, if the VPKI involves ℓ LAs, a total of ℓ pre-linkage values need to be inserted in to the CRL for each revoked device. Hence, the CRL size grows linearly with the number of *revoked devices*, not with the number of *revoked certificates*. Such mechanism is useful not only for saving bandwidth, but also because the larger the number of entries in a CRL, the higher the processing overheads for checking a certificate’s revocation status. More precisely, for each CRL entry published at time period t_s , verifying whether it covers a given certificate involves the computation of two components:

a) $ls_i(t_c)$: it takes $\ell \cdot (t_c - t_s)$ hashes to compute $ls_i(t_c)$ from $ls_i(t_s)$, where $1 \leq i \leq \ell$ and t_c is the time period in which the verification is performed. This cost may be reduced by means of pre-computation, i.e., if the vehicle always keeps the updated version of the linkage seeds, $ls_i(t_c)$, besides the original ones provided in the CRL. Nevertheless, to cope with the lack of a system-wide time synchronization (VERHEUL, 2016), devices may actually need to keep a slightly older linkage seed in memory; for example, by keeping $ls_i(t_c - \epsilon)$ for a small ϵ , it is possible to compute $ls_i(t_c)$ with only ϵ hashes.

b) $plv_i(t_c, c)$: $\ell \cdot \sigma$ encryptions are required to compute $plv_i(t_c, c)$ from $ls_i(t_c)$, since the certificate under analysis may be any out of σ that are valid in the current time period

3.4 Summary

This chapter detailed the SCMS protocol. First, it gave an overview of SCMS’s structure, describing its main entities. Then, it explained how the butterfly key expansion works, followed by key linkage process. Finally, a short analysis was made of its processing costs. The next chapter describes the three contributions of this work.

4 CONTRIBUTIONS

This chapter presents the three contributions of this work. These contributions aim to achieve the three goals mentioned on Section 1.1.

4.1 More flexible linkage and revocation procedures: *Linkage Hooks*

One limitation of SCMS's revocation procedure is that it was designed for enabling only permanent revocations. Indeed, revealing the linkage seed for a specific time period inevitably reveals all future linkage seeds and, hence, all pre-linkage values and linkage values computed from them. Therefore, once the revocation is done, the corresponding vehicle's privacy is lost. Since no “un-revoke” feature is provided, reinstating a vehicle (e.g., if it was actually revoked by mistake) requires the issuance of a new batch of pseudonym certificates.

Even though permanent revocations are likely to be the main use case in V2X, the temporary linkage and/or revocation of certificates may also be useful. For example, if a software malfunction is detected in some car models, the manufacturer might prudently

Figure 7 – Linkage tree with *linkage hooks*

Source: Made by the author.

request the temporary suspension of those vehicle’s certificates, at least until an over-the-air update is released. As another example, the MA might prefer to temporarily revoke a vehicle so it has more time to investigate that vehicle’s misbehavior reports; later, this temporary revocation can be extended, dismissed, or converted into a permanent ban. Finally, the owner of a stolen or hijacked vehicle might also request it to be temporarily tracked rather than revoked (MOALLA et al., 2012); this might be particularly sensible in case of emergency vehicles, for example.

In SCMS, the temporary suspension of a vehicle’s pseudonym certificates in a given time period would require revealing those certificates’ identifiers (e.g., their fingerprints or linkage values). This would translate, thus, to 20-40 entries (WHYTE et al., 2013) per suspended vehicle. When compared to SCMS’s permanent revocation process, which only takes one entry on the CRLs, this approach is quite inefficient.

Aiming to better accommodate such a temporary suspension process, it is possible to make a small modification on the structure of SCMS’s linkage tree. More precisely, as shown in Figure 7, it is possible to improve the flexibility of this process by adding one extra level to the tree. This extra level is composed of *linkage hooks* $\mathbf{lh}_i(t)$, which are placed between a linkage seed $\mathbf{ls}_i(t)$ and the corresponding pre-linkage values $\mathbf{plv}_i(t, \cdot)$. The derivation of $\mathbf{ls}_i(t+1)$ and *linkage hook* $\mathbf{lh}_i(t)$ from the same $\mathbf{ls}_i(t)$ consists simply in applying a different suffix to each hash function’s input (e.g., a ‘0’ suffix when deriving linkage seeds, and a ‘1’ suffix for each *linkage hook*).

With this modified structure, any pre-linkage value $\mathbf{plv}_i(t_s, \cdot)$ can be recovered from $\mathbf{lh}_i(t_s)$, whereas $\mathbf{plv}_i(t, \cdot)$ for $t \neq t_s$ remain secret. Therefore, temporary revocation and linkage can be enabled for time period t_s (and only for this time period) simply by revealing $\mathbf{lh}_i(t_s)$.

4.1.1 Possible extensions

As a side note, it is worth mentioning that the same concept can be further extended to address other use cases, such as a scenario in which only a part of the certificates from a given time period need to be linked/revoked. For example, suppose that the system’s certificates must have v different purposes, so they display distinct “key usage” fields (like in regular X.509 certificates (COOPER et al., 2008)) even though they share the same validity period. This feature might be the useful, e.g., for protecting the identity of official vehicles: whenever they do not need (or want) to be identified, they could use their regular pseudonym certificates, identical to those issued to other vehicles; however, when

they need to prove their status as official vehicles (e.g., aiming to get traffic priority), they would sign their messages with special-purpose certificates.

In this scenario, one possible approach for allowing the independent revocation/linkage of such different-purpose certificates is to create distinct linkage trees, one for each key usage. Then, the certificates sharing the same purpose could be revoked altogether as usual, by inserting ℓ *linkage hooks* (for temporary revocation) or seeds (for a permanent revocation) in a CRL. However, revoking all certificates belonging to a vehicle would lead to v times more data placed in CRLs. Conversely, a more efficient revocation can be obtained by adding one extra level to the linkage tree, with $\text{lh}_i(t)$ linking the multiple $\text{lh}_i(t, 0) \dots \text{lh}_i(t, v - 1)$. Hence, if all certificates from a given time period need to be linked/revoked, then $\text{lh}_i(t)$ would be disclosed as usual; conversely, if only certificates of a certain type needed to be linked/revoked, then the disclosure of the corresponding *linkage hook* would suffice.

4.2 Building birthday attacks against SCMS's key linkage process and preventing them using *Security Strings*

Building upon the birthday paradox, it is possible to build two different attacks against the (forward) privacy properties of SCMS's key linkage and revocation procedure. Specifically, both enable the recovery of linkage seeds were not placed in any CRL. Although attackers are unable to choose a specific target (i.e., either a vehicle or a desired time period), the probability of a successful attack grows with time, meaning that the system's privacy level drops with time. In this section, the aforementioned attacks are described and it is shown how, with little or even no overhead, this privacy degradation can be effectively avoided by modifying SCMS's linkage trees.

4.2.1 Birthday attacks on pre-linkage values

SCMS's key linkage procedure is such that, when computing multiple pre-linkage values, each LA encrypts the same plaintext using different k -bit keys. Specifically, LA_i uses $\text{ls}_i(t)$ as encryption key for calculating the c -th pre-linkage value on time period t as $\text{plv}_i(t, c) = \text{Enc}(\text{ls}_i(t), \text{la_id}_i \parallel c)$.

Unfortunately, this “multiple key, one plaintext” approach creates the conditions required for building key recovery attacks even against secure block ciphers (BIHAM, 2002;

MOUHA; LUYKX, 2015). The attack is as follows. First, the attacker picks a total of 2^n unique key values $\mathbf{ls}_i^j$, where $0 \leq j < 2^n$ and n is arbitrary. The attacker then uses these $\mathbf{ls}_i^j$ as keys to compute 2^n values of $\mathbf{plv}_i^j = \text{Enc}(\mathbf{ls}_i^j, \mathbf{la_id}_i \parallel c)$ for a target $\mathbf{la_id}_i$ and a fixed c , where $0 \leq c < \sigma$. The result is a table whose entries are $\{\mathbf{plv}_i^j, \mathbf{ls}_i^j\}$. In addition, suppose that the attacker obtains 2^m $\mathbf{plvs}$ computed by the same LA_i for the chosen index c . In this scenario, the birthday paradox dictates that, if $m + n \geq k$, then the probability that at least one of those legitimate $\mathbf{plvs}$ matches a table entry indexed by $\mathbf{plv}_i^j$ is higher than 50% (BIHAM, 2002). Whenever that happens, the attacker can recover the corresponding encryption key $\mathbf{ls}_i^j$, which should match the linkage seed $\mathbf{ls}$ effectively used to calculate the pre-linkage value in question. More precisely, this is true unless $\mathbf{ls}_i^j$ and $\mathbf{ls}$ are equivalent keys, but since secure block ciphers are designed to avoid the existence of equivalent keys, it is safe to assume that the attack will lead to $\mathbf{ls}_i^j \equiv \mathbf{ls}$.

The attack, however, does not impose many constraints on the 2^m pre-linkage values gathered by the attacker: it requires only that they are generated by the same LA_i and are associated to the same index in any given time period. Therefore, the larger the number of vehicles served by LA_i , the higher the possibility of a successful attack. The result of this attack is, thus, that the security level provided by any LA degrades with time for any initial security level k .

To illustrate such security degradation, a concrete scenario can be considered. Only in the United States, there were 272 million vehicles in operation in 2018 (STATISTA, 2018). Suppose that each of those vehicle receives certificates valid for 1 week, as proposed in (WHYTE et al., 2013). In this case, the average number of linkage values generated for the same value of c every year should be $272,000,000 \times 52 \approx 2^{34}$, independently of the actual batch size. If the same LA_i is responsible for generating all of the corresponding 2^{34} pre-linkage values, this would mean that the amount of data potentially available for the attack would be $2^m = 2^{34}$. Even if the initial security level of the system is the usual $k = 128$ bits, after one year the security against this attack would be already $k - m = 94$ bits, below the minimum of 112 bits recommended by NIST (NIST, 2015c). Adopting long-lived LAs would, thus, put the privacy of non-revoked vehicles at risk: after all, an attacker that discovers $\mathbf{ls}_i(t_s)$ is able to calculate any subsequent $\mathbf{ls}_i(t)$ for $t > t_s$. Consequently, from the attacker's perspective, the victim's pseudonym certificates are as linkable as if they were revoked in time period t_s .

Albeit powerful, in principle this attack can only be perpetrated by PCAs. Indeed, besides the LAs themselves, PCAs are the only entities that have access to pre-linkage values before a vehicle is revoked. In comparison, any other entity only learns valid pre-

linkage values after they are placed on CRLs. However, when a vehicle is revoked in this manner, the corresponding linkage seed that would be recovered in the attack ceases being secret, since, by design, the revocation process also revokes the corresponding vehicle's privacy.

In addition, PCAs are not expected to get all bits output by the underlying cipher. Instead, pre-linkage values sent to the PCA correspond to a truncated cipher output (e.g., the 8 most significant bytes when the AES block cipher is adopted (WHYTE et al., 2013)). Therefore, the attacking PCA would only be able to search for partial matches in its table, which is likely to lead to false positives. The multiple linkage seed candidates $\mathbf{ls}_i(t)$ would then have to be filtered using additional information. For example, if the pre-linkage values obtained are $\mathbf{plv}_i(t, c)$ and $\mathbf{plv}_i(t, c')$, where $c' \neq c$, the attacker can build two different tables: for the same set of 2^n keys $\mathbf{ls}_i^j$, the first table's entries would be $\{Enc(\mathbf{ls}_i^j, la_id_i \parallel c), \mathbf{ls}_i^j\}$ and the second $\{Enc(\mathbf{ls}_i^j, la_id_i \parallel c'), \mathbf{ls}_i^j\}$. The correct $\mathbf{ls}_i(t)$ would then have to be in the intersection between the sets of candidates given by each table. Another possibility consists in analyzing $\mathbf{plv}_i(t, c)$ and a subsequent $\mathbf{plv}_i(t + \alpha, c)$. This provides the attacker with two sets of candidates, one set for $\mathbf{ls}_i(t)$ and the other for $\mathbf{ls}_i(t + \alpha)$. Filtering wrong linkage seeds would then be a simple matter of checking whether or not a candidate $\mathbf{ls}_i(t + \alpha)$ can be derived by iteratively hashing a candidate $\mathbf{ls}_i(t)$. For example, for $\alpha = 1$, only correct linkage seeds are expected to satisfy $\mathbf{ls}_i(t + 1) = Hash(la_id_i \parallel \mathbf{ls}_i(t))$.

4.2.2 Birthday attacks on linkage seeds

An analogous birthday attack can be built against SCMS's forward privacy property. The goal is to expose information about revoked vehicles for time periods during which the revocation was not yet valid, i.e., to recover $\mathbf{ls}_i(t < t_s)$ from the $\mathbf{ls}_i(t_s)$ obtained from a CRL. To accomplish this, the attack exploits a vulnerability in the iterative hashing process that allows the computation of the k -bit linkage seeds. Specifically, it builds upon the fact that, while computing linkage seeds, each LA employs la_id_i as a constant prefix for each hashing operation, i.e., it makes $\mathbf{ls}_i(t) = Hash(la_id_i \parallel \mathbf{ls}_i(t - 1))$. The attack can then be mounted as follows.

The attacker starts by picking 2^n k -long distinct values $lc_{i,\alpha}(0)$, for an arbitrary value of n and for $0 \leq \alpha < 2^n/\tau$. Then, the attacker uses each of those $lc_{i,\alpha}(0)$ as the starting point of a hash chain of the form $lc_{i,\alpha}(j) = Hash(la_id_i \parallel lc_{i,\alpha}(j - 1))$, where $1 \leq j < \tau^*$ for an arbitrary τ^* . Ideally, the length of the chains created in this manner should

not be much larger than the legitimate hash chains created by the target LA_i , which is accomplished by setting $\tau^* = w \times \tau$ for a small w . The different chains can be computed either sequentially or, if multiple processing cores are available for the attack, in parallel. Assuming there is no collision between any $lc_{i,\alpha}(j)$, for all j and α , the attacker would then build 2^n hash chains at the cost of $\tau^* \times 2^n$ hashing computations. If a collision occurs, though, the attacker can simply merge the corresponding hash chains, so the resulting length would be a little larger than τ^* ; the attacker can then pick an extra $lc_{i,\alpha}(0)$, until 2^n chains are obtained.

In this scenario, the birthday paradox once again dictates that, by gathering 2^m k -long linkage seeds generated by LA_i , the attacker can match at least one of them to a previously computed $lc_{i,\alpha}(j)$ with probability higher than 50% as long as $m+n+\lg(\tau^*) \geq k$. Without loss of generality, if the match occurs between $lc_{i,\alpha}(j)$ and $ls_i(t_s)$, then a previous linkage seed $ls_i(t_s - \epsilon)$ is also a match $lc_{i,\alpha}(j - \epsilon)$ calculated by the attacker. Assuming that $lc_{i,\alpha}(j - \epsilon)$ is the actual pre-image of $ls_i(t_s - \epsilon + 1)$, rather than a second pre-image, some certificates that were not revoked before time period t_s can then be associated to their owner, concluding the attack.

This attack requires access only to the linkage seeds and their corresponding la_id_i , both of which are placed on public CRLs when a vehicle is revoked. Unlike the attack against pre-linkage values described in Sec. 4.2.1, thus, this attack can be performed by any entity, not only by PCAs. Also, the 2^m linkage seeds required in the attack can refer to any time period and to any vehicle. Hence, and even though the attacker cannot choose a specific target, the attacker's chance of success increases with the number of revocations performed involving the same LA_i . Consequently, the lifetime of any LA should once again be limited to prevent the system's initial security level k from falling below a certain threshold.

4.2.3 Protection against birthday attacks: *security strings*

One straightforward approach to address the birthday attacks described in Sections 4.2.1 and 4.2.2 is to increase the linkage seed size. For example, one could adopt 192-bit linkage seeds, thus matching the second lowest key size supported by the AES block cipher. In that case, even after gathering 2^{64} linkage seeds or pre-linkage values, the attack's cost would still be $2^{192-64} = 128$. This approach, however, does not actually eliminate the aforementioned security degradation. Instead, it only delays this issue, preventing the system from reaching a low security level too quickly. Furthermore, larger key sizes lead

Figure 8 – A more secure certificate linkage tree: adding *security strings*

Source: Made by the author.

to some undesirable overheads. Specifically, 192-bit keys would add a total of 16 bytes to each CRL entry (8 bytes per linkage seed). In addition, computing any plv would be roughly 20% more costly at vehicles and LAs, since, even if key scheduling is ignored, AES-128 and AES-192 encryptions involve, respectively, 10 and 12 rounds. Hence, this simplistic solution is actually sub-optimal both in terms of security and performance.

To address birthday attacks in a more effective and efficient manner, preventing any security degradation rather than masking it, a slightly different process for building linkage trees is proposed. It consists in using hash functions whose inputs include a *security string*, which are employed for the generation of pre-linkage values, linkage seeds, and *linkage hooks* (if the latter are present). Essentially, a *security string* I is a distinct suffix for each hash function invocation, similarly to the approach employed to prevent birthday attacks in hash-based signature schemes (LEIGHTON; MICALI, 1995; MCGREW; CURCIO; FLUHRER, 2017).

For ensuring uniqueness and better efficiency, each *security string* should comprise the fields listed in Table 1, thus taking the form $I = la_id \parallel tree_id \parallel t \parallel count \parallel depth$. This design takes into account the linkage tree's structure, where each node corresponds to a hash call, as illustrated in Figure 8. Some fields can then be simply inferred from that node's positions in the linkage tree, rather than listed explicitly in CRLs. As further discussed in Sec. 5.3, this approach reduces the total memory and bandwidth overheads incurred by *security strings*.

Using this *security string*, the linkage seeds, *linkage hooks* and pre-linkage values are computed by LA_i as follows:

- Linkage seeds:

$$ls_i(t) = Hash(ls_i(t-1) \parallel la_id \parallel tree_id \parallel t-1 \parallel 0 \parallel 0), \text{ with } ls_i(0) \text{ chosen at random.}$$

Table 1 – Components of *security strings*

Field	Suggested length (bits)	Description
<i>depth</i>	8	Node’s depth in tree (all linkage seeds are at depth 0, as shown in Figure 8)
<i>count</i>	8	Node’s index in time period and depth (starting at 0, as shown in Figure 8)
<i>t</i>	24	Time period to which the node is associated
<i>tree_id</i>	40	Tree identifier (unique per tree from a given LA)
<i>la_id</i>	16	LA’s identifier
Total	96	Whole <i>security string</i>

Source: Made by the author.

- *Linkage hooks*:

$$\text{lh}_i(t) = \text{Hash}(\text{ls}_i(t) \parallel \text{la_id} \parallel \text{tree_id} \parallel t \parallel 0 \parallel 1).$$

- Pre-linkage values:

$$\text{plv}_i(t, c) = \text{Hash}(\text{lh}_i(t) \parallel \text{la_id} \parallel \text{tree_id} \parallel t \parallel c \parallel 2).$$

It is worth emphasizing that this approach is generic enough to accommodate further changes to the linkage tree’s structure, simply by modifying the composition of the *security strings*. For example, suppose that more intermediate *linkage hooks* are added to give support to additional key usages, as discussed in Section 4.1. In this case, the *security string*’s *count* parameter is simply adjusted to the *linkage hook*’s position in the tree.

4.3 Linking certificates without Linkage Authorities: *LA-free*

In the original SCMS design, a single LA cannot identify which set of certificates belong to the same device. However, it is trivial for LAs in collusion to do so: after all, they are responsible for creating and storing pre-linkage values, so these entities can easily compute a certificate’s linkage value by combining their knowledge. This is actually the basis of SCMS’s efficient revocation process, so at first sight this might be seen as unavoidable given the scheme’s design goals. Unfortunately, though, this represents an

extra point of collusion in the VPKI, in addition to the one already presented by PCAs colluding with RAs during the creation of certificates (independently of any linkage values thereby enclosed). Even worse: as described in this section, there are many possible ways by which a dishonest RA or PCA can collude with a single LA and, as a result, gain the ability to track vehicles. The solution hereby presented, which eliminates LAs as independent entities, is aimed at reducing this attack surface while preserving the system's ability to revoke/link misbehaving vehicles in an efficient manner.

4.3.1 Privacy issues involving LAs

Besides the straightforward collusion between LAs for learning linkage values, there are a few other ways to exploit their existence as separate entities aiming to violate the system's privacy. Namely, in what follows, three possible attacks are described: replay attacks by RA, RA-LA collusion and PCA-LA collusion.

- *Replay attacks by RA*: as described in Section 3.3, in SCMS the RA is responsible for relaying the LA-generated pre-linkage values, plv_1 and plv_2 , to the PCA. Even though both pre-linkage values are encrypted, so only the PCA learns their actual values, the RA can still replay a $(\text{plv}_1, \text{plv}_2)$ pair for some (or all) pseudonym certificates from a target vehicle. If the PCA does not employ any mechanism for identifying the freshness of such (encrypted) pre-linkage values, the corresponding certificates would receive the same lv . Hence, a simple relationship is created between those certificates. Fortunately, such a relationship is obvious enough to be detectable by vehicles, which might refuse to use certificates that have the same lv : after all, for k -bit long linkage values, such collisions occur only with a negligible probability when the number of certificates is much smaller than $2^{k/2}$. Nevertheless, variants of this attack would be harder to detect. In particular, suppose the RA sends bogus pseudonym certificate requests to the PCA, containing (1) cocoon keys for which the RA knows the decryption key, and (2) the same encrypted $(\text{plv}_1, \text{plv}_2)$ pair that will be used later when requesting certificates for a target vehicle. By decrypting the bogus certificate, the RA learns the linkage value lv that is placed on the target's certificate, and can then associate that certificate to its owner. The RA might even collect raw pre-linkage values, simply by replacing plv_1 with a plv_* that was previously revealed (e.g., due to a revocation); by doing so, the companion plv_2 in the bogus certificate request can be computed as $\text{plv}_2 = \text{lv} \oplus \text{plv}_*$. Unless the PCA filters such replayed pre-linkage values, the vehicle itself would be unable

to defend from (or even detect) such attacks. Standard methods for checking data freshness, however, may encumber the batch generation process: if timestamps are used, an honest RA might be prevented from collecting many plv values from the LAs for accelerating the certificate batching generation; if data freshness is enforced via nonces, on the other hand, the PCA would need to store a large number of nonces for each LA.

- *RA-LA collusion*: by colluding with LA_1 , the RA can provide a fresh plv_1 and replay a known value plv_* from LA_2 . In that case, every linkage value $\text{lv}(t, c)$ in the target certificate takes the form $\text{lv}(t, c) = \text{plv}_1(t, c) \oplus \text{plv}_*$. Hence, even though each $\text{lv}(t, c)$ appears to be uncorrelated from the vehicle's perspective, it would be easy for RA and LA_1 to identify certificates belonging to the same vehicle: it would suffice to check whether $\text{lv}(t, c) \oplus \text{plv}_*$ results in values of $\text{plv}_1(t, c)$ that belong to a same linkage tree. Once again, preventing such attack would require the PCA to check the freshness of the $(\text{plv}_1, \text{plv}_2)$ pair.
- *PCA-LA collusion*: the PCA could collude with LA_1 to identify which values of plv_1 belong to the same linkage tree upon the generation of the corresponding pseudonym certificates. Like an RA-PCA collusion, this attack would remain undetectable by any system entity or vehicle.

4.3.2 Generating linkage values without LAs

The proposed *LA-free* approach still relies on linkage values for correlating certificates belonging to the same user, similarly to what is done in the original SCMS (see Figure 9). The linkage trees employed may encompass only linkage seeds and pre-linkage values, as in Figure 6, or also include *linkage hooks* for allowing temporary revocation of vehicles, as depicted in Figure 8. Whichever the case, however, it is the PCA who is responsible for generating its own linkage trees, playing a role similar to LA_1 in the system. Each pre-linkage value in this tree is then encrypted by the PCA with its public key, using an additively homomorphic encryption algorithm (e.g., the Paillier cryptosystem (PAILLIER, 1999)). Those pre-linkage values are associated to the same identifier $\text{tree_id}_i^{\text{PCA}}$, unique per tree, and are then sent to the RA. As a result, the RA is able to recognize which set of pre-linkage values $\{\text{plv}_{\text{PCA}}(t, c)\}$ belong to the same linkage tree, and also identify their corresponding indices in that tree, t and c . Nevertheless, the RA is unable to decrypt any given $\text{plv}_{\text{PCA}}(t, c)$, so it never learns any actual pre-linkage value.

Similarly, the RA also creates a set of linkage trees, playing a role analogous to LA_2 .

Figure 9 – Using linkage values in SCMS (top) and in the proposed *LA-free* approach (bottom)

Source: Made by the author.

Then, whenever a new batch of certificates needs to be issued, the RA selects one of its own trees, identified by $tree_id_i^{RA}$, and homomorphically adds its corresponding prelinkage values to those received from the PCA. The outcome is that the RA obtains a set of (encrypted) linkage values of the form $lv(t,c) = plv_{PCA}(t,c) + plv_{RA}(t,c)$, which can only be decrypted by the PCA. The RA then delivers to the PCA those (encrypted) linkage values, together with any other data normally included in the pseudonym certificate issuance process (e.g., cocoon keys), using SCMS's shuffling mechanism for mixing up requests from different users. Finally, the PCA creates and signs the certificates as usual, the only difference being that $lv(t,c)$ is retrieved directly from the RA's request,

instead of computed by XORing pre-linkage values provided by different LAs.

As a result of this process, even though PCA and RA create the pre-linkage values without the intervention of any LA, they have no knowledge of which pre-linkage is attached to each certificate (unless, of course, they collude). More precisely, the RA does not learn any $\text{plv}_{PCA}(t, c)$ received from the PCA, since they are random-like values encrypted with PCA's public key; hence, the RA is unable to determine $\text{lv}(t, c)$ despite the fact that its computation involves the (known) pre-linkage value $\text{plv}_{RA}(t, c)$. The PCA, in turn, is unable to determine which $\text{plv}_{PCA}(t, c)$ corresponds to a given $\text{lv}(t, c)$ received from the RA, since $\text{plv}_{RA}(t, c)$ acts as a random mask during the computation of $\text{lv}(t, c)$; therefore, assuming that the RA correctly shuffled the vehicles' requests, any received $\text{lv}(t, c)$ follows an uniform distribution from the PCA's perspective. Consequently, as in the original SCMS, the RA only knows that a given batch of certificates belongs to the same user, but has no access to the batch's contents, including the linkage values enclosed in the certificates. In comparison, the PCA knows the certificates' contents, but cannot link any information thereby enclosed to a given user; for example, it is unable to correlate a $\text{lv}(t, c)$ to its corresponding $\text{plv}_{PCA}(t, c)$ and, thus, to a specific linkage tree.

4.3.3 The *LA-free* revocation process

Revocation in the proposed *LA-free* scheme follows a process quite similar to the one originally proposed in SCMS (WHYTE et al., 2013). Namely, when the MA detects that the owner of a given certificate cert is misbehaving, it provides that certificate's linkage value, lv , to the PCA. In response, the PCA sends to the MA the identifier of the request in which cert was generated. The MA can then ask the RA for the values of plv_{RA} and plv_{PCA} employed in that request: the former is known by the RA, so its actual value can be presented together with any additional data that allows associated certificates to be revoked in a forward secure manner (e.g., a linkage seed or hook, depending on whether the revocation should be permanent or temporary); the latter, on the other hand, is homomorphically encrypted with the PCA's public key, so only this encrypted data is provided by the RA. With this information, the MA acquires from the PCA the decrypted value of plv_{PCA} , together with the corresponding linkage seed/hook that must be disclosed as part of the revocation process. Finally, the MA places those linkage seeds/hooks in a CRL, so anyone can use them in the computation of $\text{lv}(t_s, c)$, for any revoked time period t_s . The RA can also place the corresponding vehicle in a blacklist, so it cannot receive new certificates.

When compared with the original SCMS revocation procedure, the main difference in the described process is that the PCA needs to be contacted twice, the first for identifying the pseudonym certificate request and the second for the retrieval of the unencrypted value of plv_{PCA} . In addition, this process is designed to avoid the leakage of information between PCA and RA, as well as to allow extra verifications by the MA. More precisely, if desired, the MA can confirm that the correct certificate is being revoked, by checking that $\text{lv} \stackrel{?}{=} \text{plv}_{RA} + \text{plv}_{PCA}$ holds true and that the provided linkage seed/hook does produce those linkage values. Hence, if either PCA or RA sends an invalid seed/hook to the MA either due to an unintentional mistake or to malicious intent (e.g., an attempt to prevent the vehicle from being revoked), this issue can be detected. The reason is that, since the PCA (resp. RA) does not learn the value of plv_{RA} (resp. plv_{PCA}) in this process, providing a wrong value of plv_{PCA} (resp. plv_{RA}) should lead to the correct lv with negligible probability.

4.3.4 Detection of dishonest RA by MA

In principle, one possible drawback of the proposed solution is that it does not prevent a dishonest RA from providing a bogus linkage value to the PCA aiming to track devices. More precisely, suppose that the RA does not use the additively homomorphic scheme to compute the encrypted value of $\text{lv}(t, c) = \text{plv}_{RA}(t, c) + \text{plv}_{PCA}(t, c)$. Instead, it simply encrypts an arbitrary bitstring $z(t, c)$ with the PCA's public key, and then presents the resulting ciphertext in place of the correct $\text{lv}(t, c)$. By design, the PCA should be unable to distinguish a correctly computed $\text{lv}(t, c)$ from a random bitstring, since otherwise it might also be able to identify which value of $\text{plv}_{PCA}(t, c)$ was employed in the computation of $\text{lv}(t, c)$ (and, thus, to associate different requests to the same user). Therefore, such trickery would go unnoticed by the PCA, and $z(t, c)$ would be used as that certificate's linkage value. Then, it would be trivial for the RA, who knows every value of $z(t, c)$ and also the identities of the vehicles making each request, to link a pseudonym certificate to its owner.

Besides violating the users' privacy, such misbehavior from the RA might have disastrous consequences to the VPKI's revocation process. Namely, it would prevent an MA from revoking those certificates using a single pair of linkage seeds. The reason is that it is very unlikely that the set of $(\text{plv}_{RA}(t, c), \text{plv}_{PCA}(t, c))$ derived from the RA's and PCA's linkage seeds would match the arbitrary $z(t, c)$ values inserted into the certificates as linkage values. Actually, except for a negligible probability, this should only happen if every $z(t, c)$ is computed from a linkage seed, like $\text{plv}_{RA}(t, c)$, and then added to the

corresponding $\text{plv}_{PCA}(t, c)$.

This issue is taken into account in the revocation procedure described in Sec. 4.3.3, which enables the MA to identify that something is wrong when the linkage seeds provided by PCA and RA do not lead to the expected $\text{lv}(t, c)$. Indeed, when the RA forces $\text{lv}(t, c) = z(t, c)$ aiming to track vehicles, it would only pass the $\text{lv}(t, c) \stackrel{?}{=} \text{plv}_{PCA}(t, c) + \text{plv}_{RA}(t, c)$ check performed by the MA if: (1) the RA is able to provide $z(t, c) - \text{plv}_{PCA}(t, c)$ as the value of $\text{plv}_{RA}(t, c)$, as well as linkage seeds that are pre-images of such $\text{plv}_{RA}(t, c)$; or (2) $\text{plv}_{PCA}(t, c) = 0$ for every t and c , in which case the RA can simply compute $z(t, c)$ from a regular linkage tree and provide linkage seeds as usual. However, since RA never learns the value of plv_{PCA} during the pseudonym certificate issuing process, it cannot compute $z(t, c) - \text{plv}_{PCA}(t, c)$, let alone find the corresponding pre-images; in addition, the $\text{plv}_{PCA}(t, c) = 0$ condition only occurs with negligible probability, since each $\text{plv}_{PCA}(t, c)$ is the output of a hash function. Hence, it is unfeasible for a malicious RA to avert the MA's misbehavior detection whenever a certificate is revoked.

Theorem 1 summarizes this auditing procedure's security.

Theorem 1. Detection (by MA) of invalid linkage value provided by RA: *Let $\mathcal{E}$ be a homomorphic encryption algorithm, and let $\mathcal{E}(\text{plv}_{PCA})$ be the result of encrypting a secret plv_{PCA} with $\mathcal{E}$. Suppose that, given $\mathcal{E}(\text{plv}_{PCA})$, a malicious RA produces $\mathcal{E}(\text{lv}) = \mathcal{E}(z)$, for an arbitrary z . Then, that RA is able to provide an arbitrary plv_{RA} that passes the check $\text{lv} \stackrel{?}{=} \text{plv}_{PCA} + \text{plv}_{RA}$ if and only if that RA is also able to violate the confidentiality of $\mathcal{E}$.*

Proof. Since the RA ends up forcing $\text{lv} = z$, passing the check $\text{lv} \stackrel{?}{=} \text{plv}_{PCA} + \text{plv}_{RA}$ implies $z = \text{plv}_{PCA} + \text{plv}_{RA}$. Hence, if the RA is able to choose z and plv_{RA} satisfying this equation, this also means that the RA can compute $\text{plv}_{PCA} = z - \text{plv}_{RA}$ given $\mathcal{E}(\text{plv}_{PCA})$. Similarly, if the RA is able to obtain the secret plv_{PCA} from $\mathcal{E}(\text{plv}_{PCA})$, it is straightforward to pick z and plv_{RA} accordingly. $\square$

4.3.5 Detection of dishonest RA by PCA

Albeit useful, the misbehavior detection mechanism described in Sec. 4.3.4 is only applicable when a CRL is issued. Hence, it may not be enough in practice, for at least two reasons: (1) honest users are not expected to be revoked, so they could be inconspicuously tracked by a malicious RA for the entire lifetime of their certificates; and (2) some recent proposals preclude the need of CRLs, focusing instead on preventing misbehaving vehicles

from decrypting their certificates (KUMAR; PETIT; WHYTE, 2017; SIMPLICIO et al., 2018b), so RAs would be rarely (or never) scrutinized.

Fortunately, an auxiliary mechanism can be employed by the PCA for a more frequent evaluation of an RA's behavior. Namely, without loss of generality, assume that an RA should be audited periodically, after a total of n pseudonym certificates $cert_i$ (where $1 \leq i \leq n$) are issued by the PCA. Whenever this number is reached, the PCA requests (1) the sum of the n pre-linkage values generated by the RA for those certificates, denoted $\theta_{RA} = \sum_{i=1}^n(\text{plv}_{RA,i})$; and (2) the shuffled list of all encrypted $\text{plv}_{PCA,i}$ associated to those certificates — or, equivalently, a shuffled list containing the IDs of the corresponding PCA's linkage trees, as well as the indices of every $\text{plv}_{PCA,i}$ in those trees. With this information, the PCA sums up its own pre-linkage values $\text{plv}_{PCA,i}$, obtaining $\theta_{PCA} = \sum_{i=1}^n(\text{plv}_{PCA,i})$, without learning in which order each $\text{plv}_{PCA,i}$ was used by the RA. The PCA also adds together the corresponding linkage values that were inserted in the n certificates issued during that period, obtaining $\sum_{i=1}^n(\text{lv}_i)$. Finally, the PCA checks whether $\theta_{RA} + \theta_{PCA} \stackrel{?}{=} \sum_{i=1}^n(\text{lv}_i)$; the certificates were created properly only if this equality holds true.

Similarly to the MA's verification procedure, this auxiliary auditing mechanism performed by the PCA allows the MA to identify situations in which the RA responds with an arbitrary z_i instead of using $\text{plv}_{PCA,i}$ for computing the correct (homomorphically encrypted) linkage value lv_i . More precisely, by misbehaving in this manner, the RA ends up forcing $\text{lv}_i = z_i$ and, hence, $\sum_{i=1}^n(z_i) = \sum_{i=1}^n(\text{lv}_i)$. Therefore, the RA would only be able to avert detection if it provides in its response a value of $\theta_{RA*} = \sum_{i=1}^n(\text{plv}_{RA,i}) + \sum_{i=1}^n(\text{plv}_{PCA,i})$, which should be unfeasible because the RA never learns the value of any $\text{plv}_{PCA,i}$ during the pseudonym certificate issuing process. The main difference when compared with the MA's procedure is that this process prevents the PCA from learning which $\text{plv}_{PCA,i}$ is associated with each certificate, so it cannot track devices either. More precisely, the PCA only learns which pre-linkage values have already been used and, thus, can estimate how many vehicles have received the different $cert_i$. As long as the number of vehicles is large enough, however, this knowledge does not incur any privacy issue.

Theorem 2. Detection (by PCA) of invalid linkage value provided by RA: *Let $\mathcal{E}$ be a homomorphic encryption algorithm, and let $\mathcal{E}(\text{plv}_{PCA,i})$ be the result of encrypting a secret $\text{plv}_{PCA,i}$ with $\mathcal{E}$. Suppose that, given a set containing n encrypted pre-linkage values $\mathcal{E}(\text{plv}_{PCA,i})$ (say, for $1 \leq i \leq n$), a malicious RA selects some of them and responds: (1) honestly, with $\mathcal{E}(\text{lv}_i) = \mathcal{E}(\text{plv}_{PCA,i} + \text{plv}_{RA,i})$; and dishonestly, with $\mathcal{E}(\text{lv}_i) = \mathcal{E}(z_i)$, for*

arbitrary values of z_i . Then, that RA is able to provide an arbitrary θ_{RA} that passes the check $\theta_{RA} + \sum_{i=1}^n(\text{plv}_{PCA,i}) \stackrel{?}{=} \sum_{i=1}^n(\text{lv}_i)$ if and only if that RA is also able to violate the confidentiality of $\mathcal{E}$.

Proof. This can be seen as a corollary of Theorem 1. First, let H and D denote the sets of indices for which the RA responds honestly and dishonestly, respectively. The verification equation can then be divided considering those sets, so:

$$\begin{aligned} \theta_{RA} + \sum_{i=1}^n(\text{plv}_{PCA,i}) &\stackrel{?}{=} \sum_{i=1}^n(\text{lv}_i) \\ \theta_{RA} + \sum_{h \in H}(\text{plv}_{PCA,h}) + \sum_{d \in D}(\text{plv}_{PCA,d}) &\stackrel{?}{=} \\ &\sum_{h \in H}(\text{lv}_h) + \sum_{d \in D}(\text{lv}_d) \end{aligned}$$

Replacing the RA's responses for H and D , it results in:

$$\begin{aligned} \theta_{RA} + \sum_{h \in H}(\text{plv}_{PCA,h}) + \sum_{d \in D}(\text{plv}_{PCA,d}) &\stackrel{?}{=} \\ \sum_{h \in H}(\text{plv}_{PCA,h}) + \sum_{h \in H}(\text{plv}_{RA,h}) + \sum_{d \in D}(z_d) & \\ \theta_{RA} + \sum_{d \in D}(\text{plv}_{PCA,d}) &\stackrel{?}{=} \sum_{h \in H}(\text{plv}_{RA,h}) + \sum_{d \in D}(z_d) \end{aligned}$$

In principle, every $\text{plv}_{RA,h}$ and z_d are under the attacker's control, and so is θ_{RA} . However, if the RA can somehow find a value of θ_{RA} that satisfies this equation, this implies that RA's ability to compute $\sum_{d \in D}(\text{plv}_{PCA,d}) = \sum_{h \in H}(\text{plv}_{RA,h}) + \sum_{d \in D}(z_d) - \theta_{RA}$ given every value of $\mathcal{E}(\text{plv}_{PCA,d})$. In other words, this means that the RA is able to violate the confidentiality of $\mathcal{E}$ for the ciphertext $\mathcal{E}(\sum_{d \in D}(\text{plv}_{PCA,d}))$, which can be obtained from the individual $\mathcal{E}(\text{plv}_{PCA,d})$ using the additively homomorphic property of $\mathcal{E}$.

The converse implication follows an analogous reasoning: as long as the malicious RA can decrypt $\mathcal{E}(\sum_{d \in D}(\text{plv}_{PCA,d}))$, it is trivial to compute a valid θ_{RA} by combining the obtained plaintext with the previously picked $\text{plv}_{RA,h}$ and z_d . $\square$

Finally, it is noted that an analogous perusal by the RA to verify the PCA's honesty is unnecessary, since the PCA has no advantage in misbehaving during the computation of linkage values. For example, if the PCA inserts an arbitrary value z in the certificate instead of decrypting and using the lv provided by the RA, this would not reveal any information about the owner of the certificates. Instead, it would only needlessly disrupt

the revocation process in a manner that is detectable by the MA when, as discussed in Sec. 4.3.4, the latter verifies whether $lv \stackrel{?}{=} plv_{RA} + plv_{PCA}$ holds true. Actually, a similar non-issue occurs in the original SCMS protocol, in which the PCA might replace the linkage values received from the LAs by an arbitrary bitstring. Similarly to the *LA-free* solution, the PCA (1) would not gain any knowledge by doing so and (2) could be detected if the MA performs a similar verification considering the pre-linkage values provided by the LAs.

4.4 Summary

This chapter presents the three contributions of this work. The first one shows how *linkage hooks* can make SCMS's key linkage and revocation procedures more flexible by introducing a new value to its internal structure. The second contribution first shows how two types of birthday attacks can be performed against SCMS's key linkage process. Next, it presents the solution: the inclusion of *security strings*, which prevents the system's security degradation. The third contribution begins by showing some privacy issues that involve LAs, two or more entities of the SCMS protocol, followed by how their roles can be redistributed to other entities already present in SCMS. Finally, it describes how some of these entities can detect when the RA entity is being dishonest. The next chapter analyzes the aforementioned contributions in terms of performance, cost and security.

5 ANALYSIS

In this chapter, the three contributions presented in this work are analyzed in terms of performance, cost and security.

5.1 *Linkage Hooks*: Performance analysis

When compared to the original SCMS, the additional flexibility provided by *linkage hooks* comes at little cost. Specifically, checking if a certificate was temporarily revoked is just as costly as verifying if it was permanently revoked in SCMS. In comparison, one extra hash function call is required to verify whether a certificate was permanently revoked: after all, to compute pre-linkage values from a linkage seed, the vehicle first needs to compute a *linkage hook*. In practice, though, a vehicle is likely to perform this hash function call only once per time period (e.g., a week). The reason is that verifying a certificate's current revocation status requires only the *linkage hooks* themselves. Therefore, at the beginning of each time period, vehicles can use linkage seeds from CRLs to compute the current *linkage hook*, $\text{lh}_i(t)$; then, the linkage seeds can be moved to a secondary storage device, while $\text{lh}_i(t)$ is kept in fast memory for easily verifying every inbound certificate. Actually, when the number of CRL entries is small enough, the vehicle may even compute a look-up table for every revoked linkage value (i.e., using 20-40 table entries for each CRL entry) (SIMPLICIO et al., 2018a). In this case, *linkage hooks* do not need to remain in fast memory, just like would be the case with linkage seeds in the original SCMS.

5.2 *Security Strings*: Security analysis

Security strings prevent both types of birthday attacks described in Sections 4.2.1 and 4.2.2. Indeed, suppose the attacker creates a table of the form $\{h^j = \text{Hash}(\text{str}^j || I), \text{str}^j\}$, where the str^j are distinct k -long bitstrings, I is an arbitrary (but fixed) *security string*, and $0 \leq j < 2^n$ for a chosen parameter n . Once again, if some h^j matches a k -long linkage

seed $1s_i(t)$ placed in a CRL, the attacker obtains the pre-image str^j , which should match $1s_i(t-1)$ with high probability. An analogous reasoning also applies for pre-linkage values and *linkage hooks*: as long as the attacker's table also includes a given *security string* as input, collisions should reveal their corresponding pre-images in the linkage tree.

According to the birthday paradox, the probability of such collisions surpasses 50% if the attacker is able to compare 2^m k -long sample strings with each of the 2^n table entries, for $m + n \geq k$. Unfortunately for the attacker, however, a single sample is available for any *security string* I . Consequently, by design, *security strings* lead to a scenario where $m = 0$, meaning that the attack would only work if $n \approx k$. Therefore, for $k \geq 128$, building the attack table itself should be computationally unfeasible. If *tree_id* is also picked at random, the resulting *security string* would also prevent the pre-computation of (parts of) such table, similarly to what is done by salts in password hashing schemes (ANDRADE et al., 2016). Figure 10 shows a graphical representation of how adding *security strings* to SCMS's linkage trees prevent the system's (forward) privacy degradation.

Figure 10 – Long-term protection of security strings against birthday attacks

Source: Made by the author.

5.3 *Security Strings*: Performance analysis

Security strings have a negligible impact on the processing costs at LAs when generating linkage trees, as well as at vehicles when verifying a certificate’s revocation status. The reason is that the proposed length for *security strings* I , of 96 bits, is quite small when compared with the block size of modern hash functions. Therefore, the concatenation of I and other data (namely, linkage seed or *linkage hook*) still can be processed via a single call to the hash function’s underlying compression algorithm. This is the case, for example, of SHA-256 (NIST, 2015a): with block length of 512 bits, 65 of which are reserved (namely, for a 64-bit input length field, plus at least 1 bit for padding), it can process up to 447 bits of data with a single hashing call. Since linkage seeds and hooks are expected to be 128-bits long, a 319-bit or smaller *security string* does not incur any processing overhead.

In addition, the structure of the *security strings* hereby described is expected to limit the communication overhead involved in CRL distribution. Namely, the overhead of each field in a *security string* is as follows.

First, both *depth* and *count* fields can be inferred from the node’s position in the tree. Hence, neither of them need to be explicitly listed in CRLs, meaning that they incur no communication overhead.

In contrast, *la_id* and t must be explicitly included in a CRL. Since this is also the case in the original SCMS, though, these fields do not represent an extra overhead in the hereby proposed approach. Furthermore, when two or more CRL entries share the same value for those fields, a suitable data structure might be used for grouping those entries together, thus avoiding their repetition in multiple entries.

Finally, since the *tree_id* field is expected to be unique for each linkage tree created by a given LA, it must appear explicitly in every CRL entry. This field represents, thus, the only actual overhead incurred by the proposed solution when compared to the original SCMS. Since vehicle identifiers are expected to be 40-bits long (KUMAR; PETIT; WHYTE, 2017), the same length is suggested for *tree_id*, so it can index any linkage tree. Meanwhile, it prevents the system’s security degradation with at most 62.5% of the overhead resulting from the naive approach of adopting 192-bit AES keys. Nevertheless, it is possible to reduce the overhead per CRL entry depending on how this field is actually handled. For example, if *tree_id* takes the form of a serial number, its representation in a CRL could omit or compress leading zeros. Alternatively, *tree_id* might be built by the

RA itself as a concatenation of (1) its own identifier, (2) a serial number, and (3) one bit $b = 0$ or $b = 1$ for discerning between LA_1 and LA_2 , respectively. In that case: the RA can ensure that each LA's *tree_id* remains unique, simply by using different serial numbers for different vehicles; the serial number can replace the two values of *tree_id* (one for each LA) that, otherwise, would need to be placed in the CRL; and the bit b can be inferred simply by the order in which the LAs' information (i.e., *linkage hooks* or seeds) are placed in the CRL. The result would then be 40 bits per CRL entry, which means 31.25% of the 16 bytes overhead incurred in a solution relying on 192-bit keys.

As a final remark, it is interesting to note that the general approach hereby proposed for dealing with birthday attacks can be implemented without any actual overhead. Namely, this can be accomplished simply by omitting *tree_id* entirely. Even though this strategy would be unable to fully prevent birthday attacks, the resulting system's security would still be higher than what is provided by the current SCMS specification. The reason is that, since t would still be part of the hash function's input, any table mapping hashes to their pre-images would be bound to a specific time period. More precisely, suppose that some hash $h(t_1)$, computed from t_1 and part of the attacker's table, matches a legitimate linkage seed $1s_i(t_2)$. If $t_1 \neq t_2$, then the pre-images of $h(t_1)$ and $1s_i(t_2)$ inevitably differ, since the former is computed from $t_1 - 1$ and the latter from $t_2 - 1$. An analogous argument applies to a *linkage hook* $1h_i(t_2)$ and to a pre-linkage value $plv_i(t_2, c)$, the difference being that the pre-images for both would involve t_2 , whereas $h(t_1)$ would take t_1 . As a result, the system's (forward) privacy guarantees would only decrease with the number of *vehicles* whose linkage trees share the same time period, not with the size of linkage trees. This also means that, unlike the original SMCS, one distinct pre-image table is required for attacking each time period, increasing the costs of attacks targeting multiple time periods.

5.4 *LA-free*: Performance analysis

To assess the performance of the proposed *LA-free* solution, its main building blocks were benchmarked and the results were compared with those obtained with similar-purpose operations in the original SCMS. Specifically, for implementing the homomorphic encryption and decryption operations, the Paillier cryptosystem with 3072-bit keys is employed, which provides a 128-bit security level. The utilized implementation is based

on the libpaillier v0.8 (BETHENCOURT, 2010) cryptographic library. This open-source library, written in C, builds upon the GNU Multiple Precision Arithmetic Library (GNU, 2016) for delivering a reasonably efficient implementation of the Paillier algorithm among the options available in the literature. For the regular encryption and decryption in SCMS, the ECIES cryptosystem (ANSI, 2001) built upon an isochronous implementation of Curve-25519 (BERNSTEIN, 2006) is used, which once again ensures a 128-bit security level. This implementation was based on the RELIC cryptography toolkit version 0.4.1 (ARANHA; GOUVÊA, 2018), an open-source C library that is also quite optimized.

For the proposed *LA-free* solution, the time taken by the following operations were measured: key generation by PCA, which is performed only once as part of the system’s initialization; encryption of pre-linkage values by RA and PCA, which can be performed out-of-band, before pseudonym certificates are actually requested; and the homomorphic operation of pre-linkage values (by the RA) and the subsequent decryption (by the PCA), both of which are performed in-band, as part of the certificate issuance procedure. In the case of SCMS, it was considered: the (one-time) ECIES key generation by the PCA; the encryption of one pre-linkage value by each LA, which is assumed to be done out-of-band; and the decryption of two pre-linkage values by the PCA, which has to be performed in-band. The code utilized in this benchmark can be found on github (FERRAZ, 2019).

Table 2 shows the results obtained, both in cycles and in seconds, considering a single pseudonym certificate request on a small server equipped with an Intel i7-3820 CPU, at 3.60GHz. The numbers correspond to the average cost of 100,000 executions for each operation, leading to a standard deviation below 2% in all cases (except in the Paillier key generation, consequence of its primality testing, which is inherent to this process). As observed in this table, most of the processing overhead in the proposed solution can be done out-of-band and, thus, does not actually impact the latency of the certificate

Table 2 – Performance of the *LA-free* solution compared to the original SCMS

Operation		Time (cycles)	Time (seconds)	Standard Deviation	Phase
<i>LA-free</i>	Paillier key generation (PCA)	$7,290.24 \times 10^6$	$2,024.95 \times 10^{-3}$	9.68%	Bootstrap
	Encryption of each plv (PCA,RA)	151.26×10^6	42.01×10^{-3}	0.09%	Out-of-band
	Homomorphic operation (RA)	0.050×10^6	0.014×10^{-3}	0.46%	In-band
	Decryption of 1v (PCA)	144.26×10^6	47.72×10^{-3}	0.08%	
SCMS	ECIES key generation (PCA)	0.57×10^6	0.16×10^{-3}	2.00%	Bootstrap
	Encryption of each plv (LAs)	1.90×10^6	0.53×10^{-3}	1.55%	Out-of-band
	Decryption of 2 plvs (PCA)	2.71×10^6	0.75×10^{-3}	1.52%	In-band

Source: Made by the author.

provisioning process. The in-band processing of the *LA-free* solution, in turn, is dominated by the decryption procedure performed by the PCA, which is roughly $53\times$ larger than the corresponding process in the original SCMS. Nonetheless, the incurred latency remains quite small: the total time taken remains close to 40 milliseconds in the tests. Since real SCMS implementations should run on more powerful servers or specialized hardware, these numbers should be even lower in practice. Therefore, the overhead resulting from the proposed *LA-free* solution's added security should be quite small in practice.

In terms of bandwidth usage, the proposed solution also adds some overhead. Specifically, for a 128-bit security level, the data structures resulting from a Paillier encryption is expected to be 6144-bit long. In comparison, ECIES should lead to approximately 448-bit encrypted packages for the same security level, assuming 64-bit authentication tags and 128-bit pre-linkage values. On the other hand, the absence of LAs reduces the number of communications in half (namely, there are no LA-to-RA transmissions). Therefore, the net cost of the proposed solution when compared with the original SCMS is 10 KiB per pseudonym certificate, or a $6.9\times$ overhead. It is worth emphasizing, however, that this extra communication cost applies only to the RA and the PCA, which are expected to have high-bandwidth capabilities by design, whereas resource-constrained devices such as vehicles are not affected.

5.5 *LA-free*: Cost analysis

It is also interesting to examine the actual financial costs for implementing the proposed *LA-free* solution, when compared to the original SCMS. At first sight, it might seem that this proposal translates to a more expensive VPKI implementation, since extra hardware may be necessary to compensate the higher processing at the PCA, as well as the extra bandwidth costs. However, in reality an *LA-free* solution is likely to be a *less* costly VPKI. After all, the secure facilities that would be required for implementing LAs, as well as the personnel for managing them, become unnecessary. This is particularly relevant because the costs for building a PKI-like infrastructure are dominated exactly by the construction of secure premises (e.g., physical installations, access control and monitoring equipment, telecommunication and disaster recovery systems, among others) (VERISIGN, 2005). Furthermore, the main yearly costs come from the maintenance of such facilities, regular audits, and personnel expenditures (VERISIGN, 2005; ENTRUST, 2009). Hence, in practice the extra processing incurred by an additively homomorphic solution is likely to be compensated by considerable financial savings on the VPKI itself.

5.6 Summary

In this chapter, the contributions presented in this work were analyzed. First, it is showed how performance is affected with the addition of *linkage hooks*. Second, it is described how the introduction of *security strings* affects the security and the performance of SCMS. Finally, the LA-free solution is analyzed in terms of performance and financial cost. The next chapter shows some works that describe solutions related to SCMS, the protocol in which the contributions hereby presented are based on.

6 RELATED WORKS

Following the emergence of V2X, many proposals have appeared in the literature aiming to fulfill its security and privacy requirements. Notably, approaches based on pseudonym certificates seem to be promising, and are being considered in standardization efforts (ETSI, 2012; IEEE, 2017). In what follows some recent works are discussed, focusing on how they handle a vehicle’s revocation, giving an overview of the state of the art.

6.1 SEVECOM

The Secure Vehicular Communications (SeVeCom) (PAPADIMITRATOS et al., 2009a) is one of the first V2X-oriented projects discussing the provisioning and revocation of short-term certificates. The proposed solution relies on a Hardware Security Module (HSM) that generates a series of short-term public-private key pairs. Those public keys are then signed by the PCA after it authenticates the vehicle, creating the corresponding pseudonym certificates. If a misbehaving vehicle needs to be revoked, the PCA can then send a self-revocation command to the HSM, which is expected to halt using previously issued certificates. In addition, to handle attacks involving HSM-tampering, the system may also employ CRLs to revoke the corresponding vehicles. One important shortcoming of the original SeVeCom, however, is that there is no separation of rules between RAs and PCAs, meaning that the latter can easily track vehicles. This has been addressed in later architectures, including SCMS, with the introduction of RAs.

6.2 PUCA

Similarly to SeVeCom, the Pseudonym scheme with User-Controlled Anonymity (PUCA) (FÖRSTER; KARGL; LÖHR, 2016) is also heavily based on HSMs for managing long-term and pseudonym certificates. However, it focuses on providing stronger privacy guar-

antees even in case of collusion by system authorities. Specifically, pseudonym certificates from the PCA are obtained using long-term, anonymous n-show credentials (CAMENISCH et al., 2006) issued by a Long Term Certification Authority (LTCA). As a result, the number of pseudonym certificates issued to any vehicle can be controlled by the PCA, but it does not learn the identity of the vehicles requesting certificates. Revocation is then accomplished similarly to SeVeCom, by issuing a self-revocation message to the HSM. However, the strong anonymity provided by PUCA leads to a drawback in terms of revocability, since (by design) the system should be unable to identify and revoke vehicles with tampered HSMs. Conversely, the solutions hereby proposed are focused on architectures that include revocable privacy among their requirements.

6.3 SECMACE

A more recent VPKI proposal is the SECMACE (KHODAEI; JIN; PAPADIMITRATOS, 2018). The scheme focuses on protection against sybil attacks, ensuring that each vehicle receives a single pseudonym certificate for each time period even if it is allowed to contact multiple PCAs. Similarly to PUCA, SECMACE assumes vehicles use anonymous credentials to acquire pseudonym certificates from the PCA. One important difference, though, is that it supports revocable privacy via collaboration between PCAs and the vehicle’s LTCA. Albeit interesting, the overall architecture of SECMACE requires one anonymous credential to be acquired from the LTCA for each pseudonym certificate, or otherwise PCAs would be able to track vehicles. In contrast, SCMS uses the RA as a proxy to limit the number of pseudonym certificates per time period even when multiple PCAs are present; hence, it is able to tackle similar issues in a more efficient manner. Furthermore, SECMACE also assumes that vehicles have frequent access to the VPKI infrastructure. Even though this avoids the need of CRLs (and, consequently, voids the benefits of the solutions hereby proposed), fulfilling this requirement is potentially troublesome in remote areas.

6.4 IFAL, BCAM e ACPC

There are also some solutions whose goal is to reduce the size of CRLs. One example is IFAL (Issue First Activate Later) (VERHEUL, 2016), in which a vehicle can only compute a pseudonym certificate’s private key if it also receives a small piece of information known as an “activation code”. By distributing activation codes only to non-revoked vehicles,

the system ensures that certificates acquired by revoked users become virtually useless, since they cannot be used for signing messages. Hence, there is no need to keep those certificates in a CRL. An analogous but more efficient solution is proposed by BCAM (Binary Hash Tree based Certificate Access Management) (KUMAR; PETIT; WHYTE, 2017) and ACPC (Activation Codes for Pseudonym Certificates) (SIMPLICIO et al., 2018b), in which activation codes can be periodically broadcast rather than delivered one by one. More precisely, in BCAM and ACPC the activation codes are organized in a binary tree, created by a Certificate Access Manager (CAM); the periodical disclosure of some nodes of the tree (e.g., its root) then allows any non-revoked vehicles to compute its own activation code. While similar in terms of how activation codes are distributed, ACPC has at least two advantages over BCAM: better security, due to the fact that a dishonest CAM cannot track vehicles even if it colludes with other system entities (unlike BCAM); and a considerably smaller processing and bandwidth usage. All in all, IFAL, BCAM and ACPC can be seen as supplementary to this work. The reason is that their goal is to limit the growth of CRLs, so they focus on reducing the number of vehicle identifiers carried by each CRL; conversely, the focus is on how these identifiers are computed in a secure and efficient manner.

6.5 Summary

This chapter described some works that show similar solutions to the SCMS protocol. The SeVeCom and PUCA solutions both rely on HSMs and have privacy guarantees even in case of collusion, but they have a few drawbacks. SECMACE has protection against sybil attacks, but some of its requirements are not viable in some areas. Finally, the IFAL, BCAM and ACPC solutions introduce activation codes, which are bitstrings that are sent to activate the pseudonym certificates. These solutions focus on reducing the size of the CRL, and because of this, they can be seen as complementary to the contributions presented in this work. The next chapter presents the final remarks to conclude this work.

7 CONCLUSION

Vehicular communication (V2X) technologies are becoming increasingly prevalent, and are subject of multiple standardization efforts. In particular, the need for security and privacy has lead to the proposal of different strategies for the establishment of a Vehicular Public Key Infrastructure (VPKI). Among them, the Security Credential Management System (SCMS) is quite notorious for its scalable, efficient, and privacy-preserving approach for issuing and revoking pseudonym certificates.

In this work, three improvements are proposed, focusing on SCMS's linkage and revocation process. The first consists in creating a more flexible linkage process, modifying the underlying structure of SCMS's key linkage tree to enable temporary vehicle revocation. Namely, by adding one extra layer to the linkage tree, the system can more easily suspend (e.g., malfunctioning) vehicles for a short period. The second contribution refers to attacks that, building upon the birthday paradox, can degrade the system's (forward) privacy as time passes and more certificates are issued and revoked. One way to tackle this issue could be to limit the lifespan of the corresponding Linkage Authorities (LAs), thus preventing the security level from dropping below a given threshold. Instead, the proposed approach based on *security strings* completely averts this security degradation with a small overhead, or, alternatively, increases the system's security without incurring any overhead. Finally, the third improvement describes how the role of LAs can be redistributed to the PCA and RA in a secure manner, eliminating the need of maintaining LAs as independent entities. The result is improved security and lower deployment costs, with only a small increase in the total latency time for issuing pseudonym certificates. Part of the solutions proposed in this work were published in (SIMPLICIO et al., 2018a), in which building the third improvement, namely, the *LA-free* solution, was left as a topic for future work.

REFERENCES

- ALHEETI, K.; GRUEBLER, A.; MCDONALD-MAIER, K. **An intrusion detection system against malicious attacks on the communication network of driverless cars.** In: 12th Annual IEEE Consumer Communications and Networking Conference (CCNC). [S.l.: s.n.], 2015. p. 916–921. ISSN 2331-9852.
- ANDRADE, E.; SIMPLICIO, M.; BARRETO, P.; SANTOS, P. **Lyra2: efficient password hashing with high security against time-memory trade-offs.** IEEE Transactions on Computers, IEEE, Washington, DC, USA, v. 65, n. 10, p. 3096–3108, 2016. ISSN 0018-9340. See also: <http://eprint.iacr.org/2015/136>.
- ANSI. **Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, X9.63.** [S.l.], 2001.
- ARANHA, D.; GOUVÊA, C. **RELIC is an Efficient Library for Cryptography.** 2018. <https://github.com/relic-toolkit/relic>.
- BERNSTEIN, D. J. **Curve25519: new Diffie-Hellman speed records.** In: SPRINGER. Int. Workshop on Public Key Cryptography. [S.l.], 2006. p. 207–228.
- BETHENCOURT, J. **Paillier Library: libpaillier-0.8.** 2010. <http://acsc.cs.utexas.edu/libpaillier>.
- BIHAM, E. **How to Decrypt or Even Substitute DES-Encrypted Messages in 2^{28} Steps.** Inf. Process. Lett., Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 84, n. 3, p. 117–124, nov 2002. ISSN 0020-0190.
- CAMENISCH, J.; HOHENBERGER, S.; KOHLWEISS, M.; LYSYANSKAYA, A.; MEYEROVICH, M. **How to Win the Clonewars: Efficient Periodic N-times Anonymous Authentication.** In: Proc. of the 13th ACM Conference on Computer and Communications Security (CCS'06). New York, NY, USA: ACM, 2006. p. 201–210. ISBN 1-59593-518-5.
- CAMP. **Security Credential Management System Proof-of-Concept Implementation – EE Requirements and Specifications Supporting SCMS Software Release 1.1.** [S.l.], 2016.
- CERTICOM. **SEC 4 v1.0: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV).** [S.l.], 2013. <http://www.secg.org/sec4-1.0.pdf>.
- CINCILLA, P.; HICHAM, O.; CHARLES, B. **Vehicular PKI scalability-consistency trade-offs in large scale distributed scenarios.** In: IEEE Vehicular Networking Conference (VNC). [S.l.: s.n.], 2016. p. 1–8.

COOPER, D.; SANTESSON, S.; FARRELL, S.; BOEYEN, S.; HOUSLEY, R.; POLK, W. **RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**. 2008. <https://tools.ietf.org/html/rfc5280#section-4.2.1.3>.

DOUCEUR, J. **The Sybil Attack**. In: Proc. of 1st International Workshop on Peer-to-Peer Systems (IPTPS). [S.l.]: Springer, 2002.

ENTRUST. **Why outsourcing your PKI provides the best value – White Paper**. [S.l.], 2009.

ETSI. **TR 102 941 – Intelligent transport systems (ITS); security; trust and privacy management**. [S.l.], 2012.

FERRAZ, L. T. D. **Benchmark for the LA-free solution**. 2019. Available from Internet: <https://github.com/leotdferraz/lafree-benchmark.git>.

FÖRSTER, D.; KARGL, F.; LÖHR, H. **PUCA: A pseudonym scheme with strong privacy guarantees for vehicular ad-hoc networks**. Ad Hoc Networks, v. 37, p. 122–132, 2016. ISSN 1570-8705.

GNU. **The GNU Multiple Precision Arithmetic Library – GMP**. 2016. <https://gmplib.org/>.

HARDING, J.; POWELL, G.; YOON, R.; FIKENTSCHER, J.; DOYLE, C.; SADE, D.; LUKUC, M.; SIMONS, J.; WANG, J. **Vehicle-to-Vehicle Communications: Readiness of V2V Technology for Application**. [S.l.], 2014.

HU, Y. **Improving the efficiency of homomorphic encryption schemes**. PhD Dissertation (Electrical and Computer Engineering) — Worcester Polytechnic Institute, 2013.

IEEE. **IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages - Amendment 1**. IEEE Std 1609.2a-2017 (Amendment to IEEE Std 1609.2-2016), IEEE Vehicular Technology Society, p. 1–123, Oct 2017.

KATZ, J.; LINDELL, Y. **Introduction to Modern Cryptography, Second Edition**. 2nd. ed. [S.l.]: Chapman & Hall/CRC, 2014. ISBN 1466570261, 9781466570269.

KHODAEI, M.; JIN, H.; PAPADIMITRATOS, P. **SECMACE: Scalable and Robust Identity and Credential Management Infrastructure in Vehicular Communication Systems**. Trans. Intell. Transport. Sys., IEEE, Piscataway, NJ, USA, v. 19, n. 5, p. 1430–1444, 2018. ISSN 1524-9050.

KHODAEI, M.; PAPADIMITRATOS, P. **The Key to Intelligent Transportation: Identity and Credential Management in Vehicular Communication Systems**. IEEE Veh. Technol. Mag., v. 10, n. 4, p. 63–69, Dec 2015. ISSN 1556-6072.

KOBLITZ, N. **Elliptic Curve Cryptosystems**. Mathematics of Computation, v. 48, n. 177, p. 203–209, jan. 1987. ISSN 0025-5718.

KUMAR, V.; PETIT, J.; WHYTE, W. **Binary Hash Tree Based Certificate Access Management for Connected Vehicles**. In: Conference on Security and Privacy in Wireless and Mobile Networks (WiSec'17). New York, NY, USA: ACM, 2017. p. 145–155. ISBN 978-1-4503-5084-6.

LAMPORT, L. **Password authentication with insecure communication**. Commun. ACM, ACM, NY, USA, v. 24, n. 11, p. 770–772, 1981. ISSN 0001-0782.

LEIGHTON, F.; MICALI, S. **Large provably fast and secure digital signature schemes based on secure hash functions**. 1995. US Patent 5,432,852.

MCGREW, D.; CURCIO, M.; FLUHRER, S. **Hash-Based Signatures**. [S.l.], 2017. Work in Progress. Available from Internet: <https://datatracker.ietf.org/doc/html/draft-mcgrew-hash-sigs-06>.

MENEZES, A.; OORSCHOT, P. van; VANSTONE, S. **Handbook of Applied Cryptography**. [S.l.]: CRC Press, 1996. (Discrete Mathematics and Its Applications).

MILLER, V. S. **Use of Elliptic Curves in Cryptography**. In: WILLIAMS, H. C. (Ed.). Advances in Cryptology — CRYPTO '85 Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986. p. 417–426. ISBN 978-3-540-39799-1.

MOALLA, R.; LONC, B.; H.LABIOD; SIMONI, N. **Risk Analysis Study of ITS Communication Architecture**. In: 3rd Int. Conf. on The Network of the Future. [S.l.: s.n.], 2012. p. 2036–2040.

MOUHA, N.; LUYKX, A. **Multi-key Security: The Even-Mansour Construction Revisited**. In: Advances in Cryptology (CRYPTO'15). Berlin, Heidelberg: Springer, 2015. p. 209–223. ISBN 978-3-662-47989-6.

MUKUNDI, P. **List of Hash Functions (All Types with Examples) - Kifanga**. 2019. Available from Internet: <https://kifanga.com/list-of-hash-functions/>.

NHTSA. **Federal Motor Vehicle Safety Standards; V2V Communication**. [S.l.], 2017. v. 82, n. 8, 3854–4019 p.

NIST. **FIPS 180-4: Secure Hash Standard (SHS)**. [S.l.], 2015.

_____. **FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions**. [S.l.], 2015. Doi:10.6028/NIST.FIPS.202.

_____. **Special Publication 800-131A Rev. 1: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths**. [S.l.], 2015.

PAILLIER, P. **Public-key cryptosystems based on composite degree residuosity classes**. In: SPRINGER. Eurocrypt'99. [S.l.], 1999. p. 223–238.

PAPADIMITRATOS, P.; BUTTYAN, L.; HOLCZER, T.; SCHOCH, E.; FREUDIGER, J.; RAYA, M.; MA, Z.; KARGL, F.; KUNG, A.; HUBAUX, J.-P. **Secure vehicular communication systems: design and architecture**. arXiv preprint arXiv:0912.5391, 2009.

PAPADIMITRATOS, P.; FORTELLE, A. L.; EVENSSSEN, K.; BRIGNOLO, R.; COSENZA, S. **Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation**. IEEE Communications Magazine, v. 47, n. 11, p. 84–95, November 2009. ISSN 0163-6804.

PASS, R.; SHELAT, A. **A Course in Cryptography**. [S.l.: s.n.], 2010.

PETIT, J.; SCHAUB, F.; FEIRI, M.; KARGL, F. **Pseudonym Schemes in Vehicular Networks: A Survey**. IEEE Communications Surveys Tutorials, v. 17, n. 1, p. 228–255, 2015. ISSN 1553-877X.

RIVEST, R.; SHAMIR, A.; ADLEMAN, L. **A Method for Obtaining Digital Signatures and Public-key Cryptosystems**. Commun. ACM, ACM, New York, NY, USA, v. 21, n. 2, p. 120–126, feb 1978. ISSN 0001-0782.

SIMPLICIO, M.; COMINETTI, E.; PATIL, H. K.; RICARDINI, J.; FERRAZ, L.; SILVA, M. **A privacy-preserving method for temporarily linking/revoking pseudonym certificates in VANETs**. In: 17th IEEE Int. Conf. On Trust, Security And Privacy In Computing And Communications (TrustCom'18). [S.l.: s.n.], 2018. See also: eprint.iacr.org/2018/185.

SIMPLICIO, M.; COMINETTI, E.; PATIL, H. K.; RICARDINI, J.; SILVA, M. **ACPC: Efficient revocation of pseudonym certificates using activation codes**. Ad Hoc Networks, p. (in press), 2018. ISSN 1570-8705.

_____. **The Unified Butterfly Effect: Efficient Security Credential Management System for Vehicular Communications**. In: IEEE Vehicular Networking Conference (VNC'18). [S.l.: s.n.], 2018. See also: eprint.iacr.org/2018/089.pdf.

STATISTA. **U.S. car sales from 1951 to 2017 (in units)**. 2018. www.statista.com/statistics/199974/us-car-sales-since-1951/.

STEHLÉ, D.; STEINFELD, R. **Faster Fully Homomorphic Encryption**. In: ABE, M. (Ed.). Advances in Cryptology - ASIACRYPT 2010. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 377–394. ISBN 978-3-642-17373-8.

VACCA, J. R. **Public Key Infrastructure: Building Trusted Applications and Web Services**. 1st. ed. Boston, MA, USA: Auerbach Publications, 2004. ISBN 0849308224, 9780849308222.

VERHEUL, E. **Activate Later Certificates for V2X: Combining ITS efficiency with privacy**. 2016. Cryptology ePrint Archive 2016/1158. Available from Internet: <http://eprint.iacr.org/2016/1158>.

VERISIGN. **Total Cost of Ownership for Public Key Infrastructure – White Paper**. [S.l.], 2005.

WHYTE, W.; WEIMERSKIRCH, A.; KUMAR, V.; HEHN, T. **A security credential management system for V2V communications**. In: IEEE Vehicular Networking Conference (VNC'13). [S.l.: s.n.], 2013. p. 1–8. ISSN 2157-9857.