MARCOS VINICIUS MACIEL DA SILVA

ACPC: EFFICIENT REVOCATION OF PSEUDONYM CERTIFICATES USING ACTIVATION CODES

(CORRECTED VERSION)

Thesis presented to the Graduate Program in Electrical Engineering at the Escola Politécnica, Universidade de São Paulo, to obtain the degree of Doctor of Science.

São Paulo
2023

MARCOS VINICIUS MACIEL DA SILVA

ACPC: EFFICIENT REVOCATION OF PSEUDONYM CERTIFICATES USING ACTIVATION CODES

(CORRECTED VERSION)

Thesis presented to the Graduate Program in Electrical Engineering at the Escola Politécnica, Universidade de São Paulo, to obtain the degree of Doctor of Science.

Concentration area:

Computer Engineering

Advisor:

Prof. Dr. Marcos Antonio Simplicio Junior

São Paulo
2023

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: _____

Catalogação-na-publicação

Silva, Marcos Vinicius Maciel da
ACPC: Efficient Revocation of Pseudonym Certificates using Activation Codes / M. V. M. Silva -- versão corr. -- São Paulo, 2023.
127 p.

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

1.Criptografia 2.Privacidade 3.Segurança de redes 4.Comunicação veicular I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my graduate advisor, Prof. Dr. Marcos Antonio Simplicio Jr., for everything he could teach me, guiding my academic studies and inspiring me to go on.

A special thanks to my lab partners, Eduardo Lopes Cominetti and Jefferson Ricardini Oliveira, for the extensive help and insightful discussions that contributed to the development of this thesis.

And for the other colleagues and friends from LARC, whose encouragement and (mostly) funny jokes made this journey easier.

Lastly, thanks to my family and lifelong friends for the trust and support when I needed the most. Especially to my mother, without whom I would not be here today.

RESUMO

O crescimento do suporte a tecnologias de comunicação veicular (V2X) permitiu que veículos compartilhassem informações sobre condições da estrada e do próprio veículo, melhorando a eficiência e a segurança no transporte. No entanto, o uso destas tecnologias em ambientes reais pode apresentar vulnerabilidades que permitam que usuários ou outras entidades se aproveitem maliciosamente do sistema, quebrando a autenticidade das mensagens trocadas ou a privacidade dos usuários. Em especial, espera-se que o sistema seja capaz de revogar veículos maliciosos (e.g., caso enviem informações inválidas para outros veículos ou para a infraestrutura das estradas) sem permitir que veículos honestos sejam rastreados através de suas mensagens. Estas características são comumente tratadas pelo uso de sistemas de Infraestrutura de Chaves Pública Veicular (VPKI), como é o caso do Sistema de Gerenciamento de Credenciais de Segurança (SCMS), um dos principais candidatos à padronização de comunicação V2X nos Estados Unidos. Porém, enquanto o SCMS apresenta um método eficiente para prover certificados de pseudônimos, seu mecanismo de revogação pode levar a grandes Listas de Revogação de Certificados (CRLs), resultando no grande uso de bandas de comunicação e de processamento de dados. Nesta tese, é proposto um novo esquema chamado de Códigos de Ativação para Certificados de Pseudônimo (ACPC), que, integrado ao SCMS, trata o problema da revogação de certificados. A proposta é baseada em códigos de ativação, pequenas cadeias de bits enviadas periodicamente a veículos honestos, sem as quais certificados previamente emitidos a veículos revogados não possam ser utilizados. Como resultado, as entradas na CRL correspondentes a certificados revogados não precisam ser mantidos por longos períodos de tempo, reduzindo o tamanho da CRL e o seu custo de transmissão, e acelerando a verificação da validade dos certificados. E ao permitir diferentes estratégias de distribuição de códigos de ativação, o ACPC pode se beneficiar de canais broadcast ou unicast ao equilibrar entre ganhos de banda e o privacidade dos veículos na requisição. Além da descrição detalhada do ACPC, ele é comparado com outras soluções similares, como o Sistema de Transporte Inteligente Cooperativo (C-ITS), o Emita Antes e Ative Depois (IFAL), e o Gerenciamento de Acesso a Certificados baseado em Árvores Binárias de Hash (BCAM). Esta análise mostra que o esquema proposto neste documento não somente melhora a privacidade (e.g., em termos de resiliência a conluio de autoridades do sistema), como também pode levar à redução em ordens de grandeza da sobrecarga de processamento e de banda de esquemas do estado da arte.

ABSTRACT

The increased support to vehicle communication (V2X) technology allows vehicles to exchange information about road conditions and the vehicles' states, thereby enhancing transportation safety and efficiency. For broader deployment, however, such technologies are expected to address security and privacy concerns, preventing abuse by users, by the system's entities end by external attackers. In particular, the system is expected to enable the revocation of malicious vehicles (e.g., in case they send invalid information to their peers or to the roadside infrastructure) while preventing tracking of honest vehicles. These features are enabled by Vehicular Public Key Infrastructure (VPKI) solutions such as the Security Credential Management Systems (SCMS), one of the leading candidates for protecting V2X communication in the United States. While SCMS provides an efficient method for provisioning pseudonym certificates, its revocation mechanism can lead to large Certification Revocation Lists (CRLs), resulting in great bandwidth usage and processing overhead. In this thesis, we propose a novel design called Activation Codes for Pseudonym Certificates (ACPC), which can be integrated into SCMS to address this issue. Our proposal is based on activation codes, short bitstrings periodically distributed to non-revoked vehicles, without which certificates previously issued to a vehicle cannot be used by the latter. As a result, the certificates of revoked vehicles do not need to remain on the CRL for a long time, reducing the CRLs' size and streamlining their distribution and verification of any vehicle's revocation status. And by providing different distribution strategies for activation codes, we also show that ACPC can benefit from either broadcast or unicast channels by enabling different trade-offs between bandwidth savings and the requester vehicle's anonymity. Besides describing ACPC in detail, we also compare it to similar-purpose solutions such as the Cooperative Intelligent Transport System (C-ITS), Issue First Activate Later (IFAL) and Binary Hash Tree based Certificate Access Management (BCAM). This analysis shows that our proposal not only improves privacy (e.g., in terms of resilience against colluding system authorities), but can also lead to processing and bandwidth overheads that are orders of magnitude smaller than those observed in the state of the art.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ACPC	Activation Codes for Pseudonym Certificates
AES	Advanced Encryption Standard
BCAM	Binary-hash-tree-based Certificate Access Management
CAM	Certificate Access Manager
CBC	Cipher Block Chaining
C-ITS	Cooperative-Intelligent Transport Systems
CRL	Certificate Revocation List
DLP	Discrete Logarithm Problem
DR	Direct Request
DSA	Digital Signature Algorithm
DSV	Device Specific Value
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECIES	Elliptic Curve Integrated Encryption Scheme
FFC	Finite Field Cryptography
FSS	Fixed-Size Subset
HMAC	Hash-based Message Authentication Code
HSM	Hardware Secure Module
IFAL	Issue First Activate Later

IFC	Integer Factorization Cryptography
ITS	Intelligent Transportation System
KDF	Key Derivation Function
LA	Linkage Authority
MA	Misbehavior Authority
MAC	Message Authentication Code
MDC	Modification Detection Code
MitM	Man-in-the-Middle
PCA	Pseudonym Certificate Authority
PRF	Pseudo Random Function
RA	Registration Authority
RSA	Rivest-Shamir-Adleman (cryptosystem)
SCMS	Security Credential Management System
SRL	Soft-Revocation List
TE	Trusted Element
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
V2X	Vehicle to Everything
VID	Vehicle's Identifier
VPKI	Vehicular Public Key Infrastructure
VSS	Variable-Size Subset

LIST OF SYMBOLS

A	Blinded activation code
α	Number of time periods covered by an activation period
b	A generic bit
β	Number of cocoon keys in pseudonym certificates batch
c	Index of a certificate among σ in a time period
$\mathbb{C}$	Ciphertext space
cam_id	Identifier of the Certificate Access Manager
cert	A digital certificate
`code`	Activation code
D	The depth of the activation tree
$Dec(\mathcal{K}; str)$	Decryption of bitstring *str* with key $\mathcal{K}$
E	Public caterpillar key for encryption
$\hat{E}$	Public cocoon key for encryption
e	Private caterpillar key for encryption
$\hat{e}$	Private cocoon key for encryption
$Enc(\mathcal{K}; str)$	Encryption of bitstring *str* with key $\mathcal{K}$
G	Elliptic curve group generator
$Hash(str)$	Hash of bitstring *str*
I	Security string

$\mathcal{K}$	A generic key
$\mathbb{K}$	Key space
k	Key length (security level)
$\mathtt{kb}$	Batch key
$KDF\,(str)$	Key derivation function on the string str
la_id_i	Identifier of a Linkage Authority
$\mathtt{link}_\mathtt{i}$	Hash link for attacking activation codes
$\mathtt{ls}_i$	Linkage seed
$\mathtt{lv}$	Linkage value
$\mathbb{M}$	Message space
$\mathtt{meta}$	Metadata
N_d	Set of non-revoked nodes
N_r	Set of nodes in the path between a revoked node and the root
n_r	Number of revoked vehicles in the activation tree
n_t	Total number of vehicles/nodes in the activation tree
$\mathtt{node}_{depth}(count)$	The $count$-th node at level $depth$ in $\mathtt{tree}$.
$\mathcal{O}$	Point at infinity
p	Padding length when computing nodes in the activation code tree
pkg	Encrypted package
$\mathtt{plv}_i$	Pre-linkage value
r	Random value
res	Resulting shuffled batch of cocoon keys.

S	Public caterpillar key for signature
$\hat{S}$	Public cocoon key for signature
s	Private caterpillar key for signature
$\hat{s}$	Private cocoon key for signature
$\mathbb{S}$	Signature space
sig	A digital signature
$Sign\,(\mathcal{K}; str)$	Signature of bitstring str with key $\mathcal{K}$
σ	Number of certificates valid at any time period
t	Index of a time period among τ
t_s	Start time period of revocation
`tree`	A binary hash tree
τ	Number of time periods in a batch of pseudonym certificates
T	IFAL message/signature transformation
U	Public signature key
$\mathcal{U}$	PCA's public signature key
u	Private signature key
u	PCA's private signature key
V	A digital certificate
VID	Vehicle's identifier
VID_r	Revoked vehicle's identifier
VID_d	Non-revoked vehicle's identifier
$Verif\,(\mathcal{K}; str)$	Verification of signature on str with key $\mathcal{K}$

vk	Vehicle HSM's key
X	Unified public caterpillar key
$\widehat{X}$	Unified public cocoon key
x	Unified private caterpillar key
$\hat{x}$	Unified private cocoon key

CONTENTS

1 INTRODUCTION

The longstanding pursuit for Intelligent Transportation Systems (ITS) (FIGUEIREDO et al., 2001) has been leading the automotive industry to expand the variety of computing and communication capabilities in vehicles, as well as in the roadside infrastructure. In particular, the increasing support to the so-called V2X communications technologies, which includes vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) mechanisms, allows the development of many ITS-oriented applications (IYER et al., 2008) (HARDING et al., 2014). Usually, such applications involve acquiring information about road conditions (e.g., its slope, current traffic load) and about the vehicles' state (e.g., velocity, acceleration, position, distance to other cars) (PAPADIMITRATOS et al., 2009). Such information can then be shared among vehicles and also relayed to drivers, so adequate actions can be taken manually or (semi-)automatically. The expected results include a reduction on traffic congestion, transportation delays, pollution emissions and waste of fuel, as well as increase safety for both vehicles and pedestrians (FIGUEIREDO et al., 2001) (DIMITRAKOPOULOS; DEMESTICHAS, 2010).

To become largely deployed and achieve their full potential, ITS technologies need to address some important concerns, in particular those related to security and privacy (SCHAUB; MA; KARGL, 2009) (FÖRSTER; KARGL; LÖHR, 2014). Specifically, the authenticity of data exchanged via V2X is critical to prevent malicious users from abusing the system, such as forcing a target vehicle to stop by simulating a road accident ahead. It can be accomplished via digital signatures computed over the broadcast

messages, as well as with the revocation of a vehicle's certificates whenever misbehavior is detected. By relying on such Vehicular Public Key Infrastructure (VPKI), vehicles would only trust in, and act upon, authentic messages signed by non-revoked peers.

However, the use of traditional certificates would allow an eavesdropper to link messages to the same vehicle. Even if the vehicle's exact position is not part of the broadcast messages' payload, an attacker who monitors a wide area (e.g., using multiple sensors placed in strategic locations) could still infer those positions and, thus, identify the path followed by a given certificate's owner. To avoid privacy issues, there are two main solutions to vehicular networks: group signatures and pseudonym certificates. Group signatures (SALEM; IBRAHIM; IBRAHIM, 2010) (GUO; BAUGH; WANG, 2007) are used to organize several vehicles in the same group, thus messages from any group member are indistinguishable, which reduces the capability of linking messages from any specific vehicle. The downside of group signatures is related to the revocation of a single member. To provide accountability, the revocation authorities must first contact the group leader in order to open messages signed by this vehicle. Hence, the trust in the authorities must be delegated to some vehicles so that they can create groups, and only these vehicles can have the identity of the culprit disclosed. Alternatively, the usage of pseudonym certificates satisfies international standards for privacy in ITS applications (ETSI, 2021). When each vehicle carries certificates for different pseudonyms, it achieves the required unlinkability by swapping among different certificates in a same route; when legal authorities need to link messages from a vehicle, the certification authorities may disclose the different pseudonyms belonging to this vehicle. Thus, pseudonym certificates are among the most prominent solutions to VPKI.

In this work we focus on the Security Credential Management System (SCMS) (WHYTE et al., 2013) (CAMP, 2016) (BRECHT et al., 2018), which is currently

one of the leading VPKI candidate designs in the United States. Basically, SCMS proposes that each vehicle should carry multiple pseudonym certificates. Even though each individual pseudonym certificate is short-lived, the batch of certificates carried by each vehicle is expected to cover a long time range (e.g., a few years). As a result, messages signed by the same vehicle using different pseudonyms cannot be linked together, so privacy is preserved as long as its certificates are managed properly (e.g., a given certificate is not reused too often). Besides enabling the efficient generation of pseudonym certificates, SCMS also enables their revocation and linkage in case of misbehavior, thus facilitating investigations by law enforcement authorities whenever necessary. This revocation process is such that, with a single piece of information broadcast to the vehicles, multiple certificates from a revoked vehicle can be identified as invalid.

1.1 Motivation

Revocation of certificates on any traditional Public Key Infrastructure (PKI) is usually addressed via the distribution of updated Certificate Revocation Lists (CRLs): a list of identifiers for certificates that have not yet expired, but should not be considered valid anymore (e.g., because its owner's private key has been compromised). This list is signed by a trusted issuer (e.g., the authority that originally issued the certificate itself), so the authenticity of its contents can be verified.

Albeit simple, this approach has some important shortcomings when applied to the context of VPKIs. One refers to the *asynchronous nature* of CRLs, which may receive new entries and updates at any time. Traditional applications, such as web browsing, typically address this issue by directly contacting some authority and checking the current status of certificates (e.g., using the Online Certificate Status Protocol – OCSP) (SANTESSON et al., 2013). In V2X environments, however, such online verification would add too much overhead to vehicles, and may not even be possible due to limited

connectivity. Hence, when CRLs are adopted, it is almost unavoidable to observe large delays (e.g., days) until revocation updates are delivered to all vehicles.

Another issue is that each vehicle is expected to carry from 20 (BRECHT et al., 2018) to 100 (EU, 2018) pseudonym certificates for each week of operation. Therefore, if a regular CRL is employed, the number of CRL entries resulting from each vehicle revocation could be very large. Indeed, there are many proposals in the literature aiming to deal with such growth, improving the efficiency of CRL distribution (for a survey, see (NOWATKOWSKI, 2010)). Among them, the solution proposed in SCMS is one of the most effective (BRECHT et al., 2018), as it consists in issuing a large batch of certificates in advance for several future validity periods, and then inserting "linkage values" in every pseudonym certificate. As a result, all certificates belonging to the same revoked vehicle can be identified with a single CRL entry, which results in the CRLs' size growing with the number of revoked vehicles, not with the number of revoked certificates.

Although the capability of linking revoked certificates is an important feature in the context of V2X due to the large number of pseudonym certificates carried by each vehicle, it also have some drawbacks. Firstly, there is a lack of flexibility. Revoking a vehicle means also that its privacy is rescinded, since its entire set of pseudonym certificates can then be linked together. Hence, the only way to reinstate a revoked vehicle into the system is to re-provision that vehicle with new certificates. In other words, there is no "unrevoke" mechanism for situations like a vehicle being wrongly revoked, or in which the revocation occurred due to a temporary error that has been patched (KUMAR; PETIT; WHYTE, 2017).

Secondly, the gentler CRL growth comes with a cost: a CRL entry's lifespan corresponds to the duration of the *batch* of pseudonym certificates carried by the revoked vehicle. It means that those entries are not as short-lived as the pseudonym themselves, but must persist as long as the vehicle's long-term certificate. This is likely to lead to

large CRLs, because batches are expected to cover several years; for example, while some proposals in the literature suggest 1 to 3 years (WHYTE et al., 2013), others assume that each batch can enclose enough certificates to cover 30 years of operation (i.e., the whole expected lifespan of the vehicle) (KUMAR; PETIT; WHYTE, 2017). As a result, the bandwidth usage for the distribution of CRLs can become a burden if many vehicles are revoked.

And finally, the costs to verify whether a pseudonym certificate is valid also grows. SCMS was designed such that the cost of checking a certificate's revocation status is proportional to the number of entries in the CRL. For each new certificate a vehicle needs to verify, it retrieves one entry from the CRL and execute several operations over it from the time period it was inserted in to the current time period. If the computed information correlates with that of the certificate, its owner was revoked and the certificate is denied. This verification is executed for each entry in the CRL, thus the latter's growth also impacts the processing overhead at the vehicles. For example, issuing 20 certificates per week (CAMP, 2016), each vehicle would have to compute up to 20 operations per entry in the CRL every time it would need to verify a certificate.

1.2 Goals

The main goal of this work is to improve the SCMS's revocation process by focusing on reducing the size of the CRL. The CRL can be compacted by reducing the number of unexpired pseudonym certificates from misbehaving vehicles. The number of entries in the CRL can be reduced by imposing a extraction of certificates future to their issuance, which must be achieved only by honest vehicles.

1.3 Related Works

In SCMS, the verification of a certificate's revocation status is done applying one-way functions from the CRL entry to the certificate's linkage information, which results in a cost linearly dependent on the number of entries in that CRL. Therefore, keeping this number low is important not only to save bandwidth when distributing CRLs, but also to allow a faster and more energy-efficient verification of a certificate's revocation status. Unfortunately, however, the same mechanism proposed in SCMS for shortening CRLs, which associates several certificates to a same entry, also extends the lifetime of those entries: after all, linkage information placed into a CRL can only be safely removed after all certificates associated to it have expired. Consequently, even if device revocation events occur at a low frequency, CRLs may actually grow big because the corresponding entries remain in the CRL for a duration comparable to that of certificate batches (e.g., years). Actually, this issue is carried even on recently proposed improvements to SCMS's key-revocation process. For example, in (CAMP, 2016), the partial linkage information is computed using a Davies-Meyer construction (PRENEEL, 2005) aiming to avoid possible vulnerabilities related to the reversibility of encryption, a modification that does not affect the longevity of the CRLs' entries. As another example, in (SIMPLICIO et al., 2018b) the linkage information is computed with suffix-free hashes, which improves the system's resilience against birthday attacks, and another layer in the hash tree is added to simplify the temporary linkage/revocation of vehicles. Even though entries for temporarily revoked vehicles can be swiftly removed from CRLs, this facility does not apply to entries related to permanent revocations.

Differently from SCMS, in the Cooperative-Intelligent Transport Systems (C-ITS) (ETSI, 2021), the issuance of pseudonym certificates (called *authorization tickets*) does not happen prior to their use but are supplied on demand. For that, vehicles create a random key-pair and request a certificate to the corresponding public key to the Autho-

rization Authority (similar to SCMS's Pseudonym Certificate Authority) by presenting their enrollment credentials. The credentials are relayed to the Enrollment Authority (similar to SCMS's Registration Authority), and if approved, a single pseudonym credential is issued by the authentication authority. Although more recent versions of C-ITS allow requesting batches of certificates similarly to SCMS (ETSI, 2021, Sec. 6.2.3.5), to avoid linkability among pseudonyms, each certificate must be requested separately. Whenever a vehicle needs to be revoked, the pseudonym certificate is linked to the vehicle's enrollment certificate by the authorization authority, which then rejects all the following certificate requests from the vehicle. Even though this scheme reduces the overhead costs in verifying pseudonym certificates due to their short validity, it expects that vehicles should be connected to the infrastructure in operation and that they request several certificates for a given time period. Not only that, but they need to authenticate with the enrollment authority every time they intend to request new certificates, and the authentication authority is able to link pseudonyms to vehicles.

Another possible approach for reducing the size of CRLs is to rely on Bloom Filters (RAYA et al., 2007) (HAAS; HU; LABERTEAUX, 2009), a probabilistic data structure that allows the compression of lists of values. As a drawback, however, Bloom Filters introduce the possibility of false-positives, i.e., some strings that are actually not among the CRL's entries may be wrongly considered part of the CRL. In addition, for a given probability of false-positives, a Bloom Filter's size depends on the maximum number of entries supported, not on the actual number of entries in it. When compared to SCMS, where the size of CRLs depends on the number of currently revoked vehicles, this is a major drawback. For example, to support 25,000,000 revoked certificates and a false-positive rate of $\approx$0.6%, a CRL implemented as a Bloom Filter would take 32 MiB (HAAS; HU; LABERTEAUX, 2009), independent of the number of certificates actually revoked. With SCMS, however, if a single vehicle carrying 5000

certificates needs to be revoked, the CRL size would take simply 32 bytes; to revoke 5000 vehicles, thus reaching the maximum capacity of the aforementioned Bloom Filter, the CRL size would still be as small as 160 KB. Therefore, when combined with SCMS, Bloom filters are useful mostly as the data structure for storing multiple certificate identifiers at the vehicles' side (e.g., see (HAAS; HU; LABERTEAUX, 2011)), an approach that is orthogonal to our proposal.

A more promising approach, which is also adopted in this work, is to prevent a revoked vehicle from accessing its pseudonym certificates. This is the basic idea behind the Issue First Activate Later (IFAL) scheme (VERHEUL, 2016) (VERHEUL; HICKS; GARCIA, 2019). In IFAL, non-revoked vehicles can periodically query the system for the "activation codes", small pieces of information required to compute the private keys for their own certificates. As a result, the identifiers for revoked certificates that have been activated need to remain in the CRL only until their corresponding expiration dates, since subsequent certificates issued to the same vehicle cannot be activated. Albeit interesting, IFAL suffers from a significant issue: since the Pseudonym Certificate Authority (PCA) communicates directly with the vehicles, it can link the pseudonym certificates it issues to the corresponding vehicle's enrollment certificates, even without colluding with any other entity. The only protection offered by IFAL against such linkability risk is that it requires the PCA to delete the information produced during the certificate issuance process (VERHEUL, 2016, Sec. 3.2). However, since such requirement can be easily ignored by a dishonest PCA, IFAL ends up offering a lower level of privacy than SCMS itself, hindering a possible integration between the two solutions.

The Binary Hash Tree based Certificate Access Management (BCAM) scheme (KUMAR; PETIT; WHYTE, 2017) is another scheme that uses activation codes for reducing CRL sizes. Unlike IFAL, however, BCAM was designed to interoperate with the SCMS architecture, inheriting its ability to protect the privacy of honest users

against a dishonest PCA as long as it does not collude with other system entities. More precisely, in BCAM the Registration Authority (RA) does not simply relay the PCA-encrypted certificate batches to the requesting vehicle, but first sends them to a Certificate Access Manager (CAM). The CAM then encrypts those batches using a symmetric cipher, so they can only be decrypted by vehicles that are able to compute the corresponding activation codes, called device specific values (DSVs). Instead of requiring vehicles to explicitly request their DSVs, though, the CAM builds a binary hash tree whose leaves allow the DSVs to be computed. The CAM then broadcasts the nodes of the tree that allow non-revoked vehicles (and only them) to recover their own DSVs. Even though BCAM does improve SCMS's revocation process, and the distribution of activation codes is considerably more efficient than the one in IFAL, it can be further improved in terms of privacy and efficiency. Namely, one drawback of BCAM's design is that it creates an extra point of collusion in the SCMS architecture: the CAM, like the RA, learns which batch of PCA-encrypted certificates belongs to a same vehicle; consequently, the CAM can collude with the PCA to violate those certificates' unlinkability and, hence, the users' privacy. In addition, during the butterfly key expansion, the CAM encrypts certificates that have already been encrypted by the PCA; therefore, this process can be optimized if those two encryption processes are combined into one. Both issues are addressed in the proposed solution.

1.4 Contribution

Aiming to tackle this CRL expansion issue, we improve SCMS's revocation process with the adoption of activation codes: small pieces of information without which pseudonym certificates previously issued become useless. Hence, by preventing revoked vehicles from obtaining those codes, the entries associated with those certificates can be safely removed from CRLs. The proposed design, named Activation Codes for Pseudonym Certificates (ACPC), builds upon previous works such as Issue

First Activate Later (IFAL) (VERHEUL, 2016) and Binary-hash-tree-based Certificate Access Management (BCAM) (KUMAR; PETIT; WHYTE, 2017). Nevertheless, it addresses privacy and performance issues in both solutions, leading to a more robust key revocation process for V2X communications. Albeit ACPC relies on the broadcast of activation codes, it can also benefit from the unicast distribution. We extend our solution by discussing three different request policies with different trade-offs between privacy and bandwidth savings.

1.5 Outline

The remainder of this thesis is organized as follows. Chapter 2 presents the cryptographic concepts and notation used in the entire thesis. Chapter 3 describes the main vehicular public key infrastructure schemes from the literature, later compared with our proposal. We then present the proposed ACPC solution in Chapter 4. Chapter 5 discusses the security of ACPC's certificate issuance and revocation procedures. Chapter 6 compares our solution with related works in terms of security and efficiency, from a theoretical and experimental perspectives. Finally, Chapter 7 concludes the thesis.

2 BASIC CONCEPTS AND NOTATION

In this chapter, we review the building blocks and notation applied in the construction of our solution. Our proposed pseudonym certification method ACPC is based on the Security Credential Management System (SCMS) (further detailed in Chapter 3.1), which relies on some cryptographic schemes. Thus, we start our discussion with the basic definition of symmetric cryptography algorithms, namely a block cipher and cryptographic hash function. We follow defining the mathematical properties behind elliptic curve cryptography and the asymmetric cryptographic protocols built over these constructions: encryption and signature systems. We note that all definitions are quite standard, and a reader with a background in cryptography may skip the following sections. Unless explicit, the definitions can be found in (KATZ et al., 1996). Our tests and benchmarks were constructed using standardized methods, and the specific schemes are presented in Table 1.

Table 1: List of cryptographic methods used in the implementation

Cryptographic function	Algorithm	
Block cipher	AES-CBC	(NIST, 2001a) (NIST, 2001b)
Hash function	SHA256	(NIST, 2015)
Digital signature	ECDSA	(NIST, 2013)
Asymmetric encryption	ECIES	(IEEE, 2004a)

Source: the author.

2.1 Cryptographic Background

Cryptography is the study of mathematical techniques to guarantee secure information management. The goals for its application depend on its security requirements since they can vary from protecting communication data to authorization of personnel. Usually, it provides at least one of the main security services:

- Confidentiality/Privacy: keeping information secret from unauthorized parties.
- Data integrity: detection or prevention of illegal data modification.
- Authentication: identification of the source or attestation of an identity.
- Non-repudiation: prevention of denial of a commitment in case of dispute.

Cryptographic primitives are employed to address these requirements, such as hash functions for data integrity, encryption schemes for confidentiality, signature schemes for authentication, and non-repudiation. They are usually categorized by its key dependency: cryptographic tools that do not depend on keys are called *unkeyed* primitives; when all parties use the same key, or the same key can be derived from information from all parties, they are called *symmetric-key* primitives; and schemes that only one party has access to some private function, while others only operate on a public function, are called *asymmetric-key* or *public-key* primitives.

The strength of these algorithms is given by the *security level*, which represents the number of bits of a random sample that an attacker must guess correctly in order to "break" the system. For unkeyed primitives, it usually depends on their output size or inner components, while for keyed primitives (both symmetric and asymmetric), it usually relates to the key size.

In this work, most unkeyed and symmetric-key primitives are treated as black-box implementations, and their inner workings are not required for the execution. How-

ever, some mechanisms from public-key primitives are explicit, which requires some background on elliptic curve cryptography.

2.2 Elliptic Curve Cryptography

We now present some group theory concepts used in the remaining of this work. Unless explicit, the definitions in this section were transcribed from (SHOUP, 2009).

Definition 1 (Abelian Group)**.** *An* abelian group $(\mathbb{G}, \star)$ *is a set* $\mathbb{G}$ *with a binary operation* $\star$ *such that the following properties hold:*

- *Closure:* $\forall A, B \in \mathbb{G}, (A \star B) \in \mathbb{G}$.
- *Associativity:* $\forall A, B, C \in \mathbb{G}, A \star (B \star C) = (A \star B) \star C$.
- *Identity element: the unique element* $I \in \mathbb{G}$ *is called the* identity element*, such that* $\forall A \in \mathbb{G}, A \star I = I \star A = A$.
- *Inverse element: for each element* $A \in \mathbb{G}$*, exists an element* $\overline{A} \in \mathbb{G}$ *called the* inverse of A *such that* $A \star \overline{A} = \overline{A} \star A = I$.
- *Commutativity:* $\forall A, B \in \mathbb{G}, A \star B = B \star A$.

Although any operation can be defined in an abelian group, it is usually defined as the *addition* or the *multiplication* operation. Thus, the "$\star$" operator is replaced by the addition "+" (resp., the multiplication "$\cdot$") operator, the identity element I by $0_{\mathbb{G}}$ (resp., $1_{\mathbb{G}}$), and the inverse $\overline{A}$ by $(-A)$ (resp., A^{-1}). The formal representation of a group is $(\mathbb{G}, \star)$, but when the operation $\star$ is well known for the set, it can be omitted, and the group is represented only by the set $\mathbb{G}$. For generic abelian groups, the addition notation is more common and usually omitted.

In several occasions, it is useful to define the composition operation, in which $\star$ is computed several times over the same element. For the addition (resp., multiplication)

operation, the composition is usually represented by the multiplication (resp., exponentiation): to compose n times on the element $A \in \mathbb{G}$, $n \cdot A = nA = \sum_{i=0}^{n-1} A$ (resp., $A^n = \prod_{i=0}^{n-1} A$). The composition operation can also be applied to provide additional structure inside a group in the form of a cyclic group.

Definition 2 (Cyclic Group)**.** *An additive (resp., multiplicative) group $\mathbb{G}$ is called a* cyclic group *if there is an element $G \in \mathbb{G}$ such that $\forall A \in \mathbb{G}$, there is an integer n such that A can be computed from the n-multiplication (resp., n-power) of G, i.e., $nG = A$ (resp., $G^n = A$). The element G is called a* generator *of $\mathbb{G}$, and the group $\langle G \rangle = \mathbb{G}$ is* generated *by G.*

The *order* of a group is the number of elements in its set and is represented by $|\mathbb{G}|$. In some cases, we only intend to apply the group operation in a subset $\mathbb{G}_1$ of smaller order than $\mathbb{G}$ ($\mathbb{G}_1 \subset \mathbb{G}$). If every property of an abelian group holds to this subset $\mathbb{G}$, we call $\mathbb{G}_1$ a *subgroup* of $\mathbb{G}$. For some element $G \in \mathbb{G}$, we can define $\langle G \rangle \subset \mathbb{G}$ as the *subgroup generated by G*. The order of the generator G is defined by the order of the subgroup $\langle G \rangle$, and is represented by $\#G$.

In some cases, the set may present both operations in its structure (namely, the addition and the multiplication), which can classify a field.

Definition 3 (Field)**.** *A* field $(\mathbb{F}, +, \cdot)$ *is a set $\mathbb{F}$ with two binary operations $+$ and $\cdot$ (called addition and multiplication) such that the following properties hold:*

- $(\mathbb{F}, +)$ *is an abelian group with identity element* 0.
- $(\mathbb{F}, \cdot)$ *is an abelian group with identity element* 1.
- *Distributivity (of multiplication):* $\forall A, B, C \in \mathbb{F}, (A + B) \cdot C = A \cdot C + B \cdot C$.

If the number of elements n in the field is limited, it is called a *finite field* or *Galois field* $\mathbb{F}_n$. The order n of the finite field must be a prime or a power of a prime ($n = p^b$, for some $b > 0$), and this prime p is the characteristic of the field.

Definition 4 (Characteristic of a finite field)**.** *If $\mathbb{F}$ is a field, there is a positive integer p such that $p \cdot A = 0$ for all $A \in \mathbb{F}$. The smallest such integer p is called the* characteristic *of the field $\mathbb{F}$. If there is no such integer p, the set $\mathbb{F}$ is said to have characteristic 0, and $\mathbb{F}$ is not a field.*

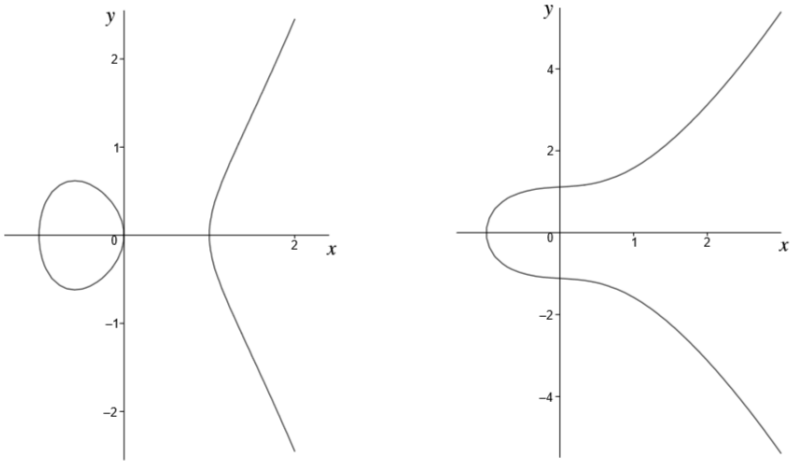
An elliptic curve is the set of solutions in some defined field to a cubic polynomial equation in two variables. As a plane curve, elliptic curves have the shape of the graphs presented in Figure 1. Formally:

Definition 5 (Elliptic Curve)**.** *An* elliptic curve *EC over a field $\mathbb{F}$ is the set of solutions $(x, y) \in \mathbb{F}$ of the equation:*

$$EC/\mathbb{F} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6; \qquad a_1, a_2, a_3, a_4, a_6 \in \mathbb{F} \qquad (2.1)$$

, together with the "point at infinity" O. Equation 2.1 is called the generalized Weierstrass form.

Figure 1: Example of elliptic curves in the real plane



(a) $E_1 : y^2 = x^3 - x$ (b) $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

Source: Hankerson, Menezes and Vanstone (2004, p. 77)

If the curve is nonsingular (i.e., there are no repeated roots to the Weierstrass equation), the set of points of an elliptic curve is an additive abelian group with identity O (HUSEMÖLLER, 2004, Remarks 2.1 and 5.3). The group operation is called point

addition, defined by the chord-tangent law of composition, and the composition of point additions is called *scalar multiplication* (WEIL, 1928), and they are the basis for designing some standardized asymmetric cryptographic methods.

The security of traditional asymmetric-key cryptographic primitives is based on mathematical problems which are easy to verify but hard to compute. Namely, the Integer Factorization Cryptography (IFC), which is based on the hardness of finding primes that, multiplied, result in a large public number (e.g., Rivest-Shamir-Adleman – RSA – cryptosystem (RIVEST; SHAMIR; ADLEMAN, 1978)); and the Finite Field Cryptography (FFC), which is based on the hardness of finding the number of compositions in an abelian group, named the discrete logarithm (e.g., Digital Signature Algorithm – DSA (NIST, 2013)).

Definition 6 (Discrete Logarithm Problem (DLP))**.** *Consider a (multiplicative) group* $\mathbb{G} = \langle G \rangle$*. Given* $P \in \mathbb{G}$*, find an integer* x *such that* $P = G^x$*.*

Elliptic Curve Cryptography (ECC) is an adaptation of the FFC using elliptic curve groups, and its security reduction is usually based on a specific instance of the DLP.

Definition 7 (Elliptic Curve Discrete Logarithm Problem (ECDLP) (MILLER, 1986))**.** *Consider an elliptic curve EC defined over the field* $\mathbb{F}_n$*. Given two points* $P, G \in EC$*, find an integer* x *such that* $P = xG$*, if such* x *exists.*

To ensure the same security level, schemes based on the DLP usually require smaller key sizes and have faster execution time than the ones based on integer factorization. And albeit both FFC and ECC are based on the DLP, there are specific attacks on the structure of FFC (e.g., index calculus (ADLEMAN, 1979)), which do not apply to ECC (SILVERMAN; SUZUKI, 1998) (MILLER, 1986). Thus, FCC usually requires larger parameters than ECC. Table 2 presents a comparison of the minimum key-lengths for IFC, FFC and ECC.

Table 2: Key sizes (in bits) for protocols based on the Integer Factorization Cryptography (IFC), Finite Field Cryptography (FFC) and Elliptic Curve Cryptography (ECC)

Security level	IFC (key size)	FFC (public key; private key)	ECC (key size)
≤ 80	1024	1024; 160	160 – 223
112	2048	2048; 224	224 – 255
128	3072	3072; 256	256 – 383
192	7680	7680; 384	384 – 511
256	15360	15360; 512	512+

Source: adapted from Barker (2016).

In this work, tests and implementation of elliptic curve protocols employ the Curve25519 (BERNSTEIN, 2006), which presents a security level of 128 bits and is one of the fastest curves free of patents.

2.3 Cryptographic Hash Functions

One-way functions take a distinctive role in cryptography because they make it impracticable for any party to compute the preimage of the public output of specific functions. Such usage is essential so that an adversary cannot compute private keys from the public parameters in most public-key cryptographic schemes. Informally, a function is called a one-way function if it is easy to compute the image from any element from the domain, but it is computationally infeasible to compute the inverse from most elements of the codomain.

Cryptographic hash functions are a particular class of one-way function that maps arbitrary-length bitstrings to another bitstring of fixed length. These functions are widely adopted with digital signature schemes to get a digest from a larger message before signing it. They are also used to assert data integrity since any alteration in the data will result in a different output (called simply *hash*).

Definition 8 (Hash function)**.** *A* hash function *is, in general sense, a function Hash which has at least two properties:*

- compression*: Hash maps an input str of arbitrary finite bitlength to an output Hash(str) of fixed bitlength n, i.e., Hash* : $\{0,1\}^* \rightarrow \{0,1\}^n$.

- ease of computation*: given Hash and str, it is easy to compute Hash(str).*

When considering cryptographic hash functions*, other properties may also be required:*

- (first) preimage resistance*: for all specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find a preimage str′ such that Hash(str′) = y when given any y for which a corresponding preimage is not previously known.*

- second preimage resistance*: it is impossible to find any second input which has the same output as any specified input, i.e., to find a second preimage str″ ≠ str′ such that Hash(str″) = Hash(str′).*

- collision resistance*: it is computationally infeasible to find any two inputs which hash to the same output, i.e., to find str′ and str″ such that Hash(str′) = Hash(str″).*

Another classification of hash functions reflects the properties the function presents. We call a *one-way hash function* a hash function that presents both the preimage resistance and second preimage resistance properties. We call a *collision-resistant hash function* a one-way hash function that also presents the collision resistance property. In the rest of this thesis, when not specified otherwise, a hash function will be considered a *collision-resistant hash function*.

Although considered a symmetric-key cryptographic protocol, hash functions can be *keyed* or *unkeyed*. The computation of a hash function output does not require a key as input, but some applications use a key as an additional input. The key is affixed to the input message so that the hash function with a given message will output two

different hashes when using two different keys. This process is related to the selection of an element of a family of hash functions, in which the family is represented by the hash function protocol and the index of the element is the key.

One common use of hash functions is in the construction of *key derivation functions (KDF)*. Given the unpredictability of the hash function output, they can be used to modify the key material for some other cryptographic algorithm to a more uniform distribution of bits. In special, its application is recommended if the key material has some predetermined format which could result in "weak bits" (i.e., bits that could be guessed by an attacker with greater probability, thus reducing the effective security of the key). Besides, they can also be used in protocols that require several operations using the same key so that the key can be updated after a determined number of steps.

Another use of hash functions is function to protect against code modification in constructions of *modification detection codes (MDC)* and *message authentication codes (MAC)*. MDC is a direct usage of an unkeyed hash function to prove that some data has not been modified after it was provided. Any modification to the original data will produce a different MDC than originally published, thus asserting its integrity. Alternatively, a MAC provides not only data integrity but also the authenticity of its source between two peers by hashing the data together with a shared authentication key.

For implementation purposes of this work, we use the hash function SHA256 and the hash-based message authentication code (HMAC) (NIST, 2008).

2.3.1 Birthday Attacks and Security Strings

Birthday attacks are a class of problems that can be applied to any hash function. These attacks aim to find sample inputs to hash functions that output to the same value (i.e., a collision). They receive this name based on the birthday problem, which defines the probabilities of finding two people with the same birthday in a group of random

people. This problem is commonly named a paradox because of the rapid growth of the probability after it reaches 50%: for a year of 365 days, it only requires a group of 23 people to get a probability of 50%, and 70 people to a probability of 99.9%.

For cryptographic hash functions, a generic birthday attack is to accumulate enough input values to a hash function to get a high probability of finding a collision. Since this number is approximate to the square root of the possible outputs, they are also characterized as a square-root attack. Considering a security level k, a collision-resistant hash function has output blocks of at least $2k$ bits. Thus, an attacker need access to $\sqrt{2^{2k}} = 2^k$ hash output values to get a collision with high probability. Although it is usually hard to obtain by itself, some applications may publish a large number of values, making the attack easier.

One way to avoid the disclosure of many hash output values is to choose different hash functions. Instead of using different algorithms, it is possible to obtain different hash outputs using the same algorithm but appending some information to the input specific to the running instance. The appended information is called a *security string* (LEIGHTON; MICALI, 1995), that uniquely define the instance of the hash function running. The construction of a security string is specific to the application, but it usually contains some information to identify the entity that provides the hash and some public information to index the input.

In Appendix A we present a scenario in which an attacker attempts a birthday attack on BCAM (KUMAR; PETIT; WHYTE, 2017), whose sources of these input values are twofold: the disclosed activation codes and the attacker chain.

2.4 Encryption

Generically, an encryption scheme (sometimes referred as a cipher) is a set of protocols capable of providing data confidentiality, i.e., of keeping the content of the infor-

mation from all but the authorized users. It is based on the reversibility of a function, which requires a key to the correct execution. The process called *encryption* receives the plaintext and an encryption key to transform it into a ciphertext. The inverse process, called *decryption*, receives the ciphertext and the corresponding decryption key to transform it in the original plaintext. Confidentiality is achieved when the ciphertext is indistinguishable from a random string, and there is no way to correlate a ciphertext to some plaintext if the decryption key is not available. Formally:

Definition 9 (Encryption Scheme)**.** *Let* $\mathbb{K}$*,* $\mathbb{M}$ *and* $\mathbb{C}$ *denote, respectively, the* key space*, the* message space *and the* ciphertext space*. The elements* $\mathcal{K} \in \mathbb{K}$*,* $str \in \mathbb{M}$ *and* $pkg \in \mathbb{C}$ *are called, respectively, a* key*, a* message *and a* ciphertext*; and the keys can be divided into the encryption key* $\mathcal{K}_e$*, and the decryption key* $\mathcal{K}_d$*. The application* $Enc\left(\mathcal{K}_e; str\right) \rightarrow pkg$ *is called* encryption*, and the application* $Dec\left(\mathcal{K}_d; pkg\right) \rightarrow str$ *is called* decryption*. The set of algorithms* $\left(Enc\left(\mathcal{K}_e; \cdot\right), Dec\left(\mathcal{K}_d; \cdot\right)\right)$ *is called an* encryption scheme *iff.* $Dec\left(\mathcal{K}_d; Enc\left(\mathcal{K}_e; str\right)\right) = str$*.*

A cipher may be either a symmetric-key scheme (when it is easy to compute the decryption key from the encryption key) or a public-key scheme (with no easy transformation between the key-pair), usually depending on the underlying computational problem. As both kinds are presented in SCMS, we further detail the symmetric-key cipher in Section 2.4.1 and the asymmetric one in Section 2.4.2.

2.4.1 Symmetric-Key Cipher

A symmetric-key cipher is an encryption scheme in which it is easy to compute the decryption key from the encryption key (e.g., if both keys are the same). This primitive is used when both the sender and the recipient of some message know the shared key or when only one user encrypts and decrypts its messages. There are two classes of symmetric-key ciphers: block cipher and stream cipher.

A *block cipher* is an encryption scheme that encrypts a fixed-size message. Whenever a message is smaller than the block size, it is necessary to append a predefined bit sequence called *padding* to round the number of blocks (e.g., (NIST, 2001b)). For messages larger than the block size, the cipher may use some *mode of operation*: a protocol that defines how to pad and split the message into blocks that can be used by the block cipher, and how the block key is derived from the cipher key. Symmetric-key block ciphers are one of the most fundamental cryptographic primitives since it is possible to create other functionalities, such as pseudorandom number generators, message authentication codes, and hash functions. It is an important primitive to SCMS, which employs the Advanced Encryption Standard (AES) in the Cipher Block Chaining (CBC) mode of operation.

Alternatively, a *stream cipher* is an encryption scheme that encrypts messages of arbitrary sizes. They are considered block ciphers with a block size of 1 bit, with a method of key derivation to produce a keystream of any size. It is not used in this work, but an interested reader can find more information in (KATZ et al., 1996).

2.4.2 Asymmetric-Key Cipher

Unlike its symmetric counterpart, an asymmetric-key (or public-key) cipher is an encryption scheme on which it is not possible to compute the decryption key from the encryption key without some secret trapdoor. In this primitive, the encryption key is publicly disclosed, while the decryption key is kept private. Hence, the main usage of such a scheme is to provide any user the capability of encrypting strings but reserving the decryption of such strings to an authorized user. It is important to note that an asymmetric-key cipher provides data confidentiality without origin authentication or integrity, which can be achieved by coupling with a digital signature and a Message Authentication Code (MAC).

Public-key cryptography is usually slower than symmetric-key, so asymmetric-

key ciphers are usually used to encrypt small messages or the shared key for some symmetric-key encryption. The SCMS employs the Elliptic Curve Integrated Encryption Scheme (ECIES) (IEEE, 2004a), which computes a shared key for a symmetric-key cipher (namely, AES-CBC) and produces a MAC (namely, hash-based MAC – HMAC) for origin authentication and integrity.

2.5 Digital Signature

A digital signature scheme is a set of algorithms capable of binding an identity to some piece of information. It provides both authentication and non-repudiation; thus, it guarantees that the owner of the public key has signed the message, and does not allow the signer to deny having signed it. The process of *signing* receives the message and the signature key, which is private to the signer, to return a signature on the message. The *verification* process receives the signature, the message, and the verification key, which in turn is public, and accepts it if and only if the signature is valid. The existence of a private signature key and public verification key intrinsically requires the signature scheme to be an asymmetric-key cryptographic protocol whose trapdoor is related to the signature key. Formally:

Definition 10 (Signature Scheme)**.** *Let* $\mathbb{K}$*,* $\mathbb{M}$ *and* $\mathbb{S}$ *denote, respectively, the* key space*, the* message space *and the* signature space*. The elements* $\mathcal{K} \in \mathbb{K}$*,* $str \in \mathbb{M}$ *and* $sig \in \mathbb{S}$ *are called, respectively, a* key*, a* message *and a* signature*; and the keys are divided into the signing key* $\mathcal{K}_s$ *and the verification key* $\mathcal{K}_v$*. The application* $Sign\left(\mathcal{K}_s; str\right) \rightarrow sig$ *is called* signature*, and the application* $Verif\left(\mathcal{K}_v; str, sig\right) \rightarrow \{0, 1\}$ *is called* verification*. The set of algorithms* $\left(Sign\left(\mathcal{K}_s; \cdot\right), Verif\left(\mathcal{K}_v; \cdot\right)\right)$ *is called a* signature scheme *iff.* $\{Sign\left(\mathcal{K}_s; str\right) \rightarrow sig \wedge Verif\left(\mathcal{K}_v; str, sig\right) = 1\}$.

For most applications, the message to be signed can be arbitrarily large. In order to sign such messages, a collision-resistant hash function may be used to compress

the original message to another message in the *signing space* (i.e., the message space in which the signing process transformations are applied). The collision resistance protects the signer so that it is infeasible to find messages that hash to the same digest (i.e., so that one signature could be valid for more than one message), especially if the messages are not provided by the signer him/herself.

In this work, we consider digital signature schemes with *appendix*, in which the message must be sent together with the signature. However, some schemes provide *message recovery*, which allows the original messages to be recovered from the signature, and the verification process does not require the message as input.

2.5.1 Public-Key Infrastructure

Signatures provide message authentication by their signer, thus providing irrevocable information that the owner of the verification key has signed the message. However, it is not enough to guarantee that the keys' owners are indeed related to a real-world identity. Thus, an entity can obtain a *digital certificate* that binds the verification key to the identity.

Besides linking the public key to the entity's identity, digital certificates also usually contain information about the certificate authority that issued that certificate, the validity period of the keys, and how the keys can be used by that party (e.g., if that key can be used to sign other certificates). In order to ensure the correctness of this information, an entire Public Key Infrastructure (PKI) is required. Formally, a PKI is the set of hardware, software, people, policies, and procedures that are needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography (SHIREY, 2007). Although a traditional PKI is based on X.509 certificates (BOEYEN et al., 2008), alternative certificates are used for specific domains (e.g., IEEE 1609.2 certificates for vehicular wireless environment (IEEE, 2016)).

2.5.2 Implicit Certificates

While traditional signature schemes are most commonly used by the certificate authorities to issue a digital certificate, it is also possible to create the end-user keys by implicitly certifying them. This idea originated from identity-based certificates (GÜNTHER, 1990), which allows the users' public keys to be derived from public attributes that uniquely define them (e.g., name, e-mail address). Only then, the certificate authorities could generate the corresponding private key from this public information and sent to their owner using a secure channel. However, this schemes resulted in the disadvantage of requiring key escrow, i.e., the certificate authorities are privy of the user private keys, since they were responsible for its generation.

In order to solve the key escrow property from identity-based certificates, implicit certificates (CERTICOM, 2013) allowed users an additional secret input in the private key generation. The implicit certificates are not actually signatures on the users' public keys, but values from which other entities could derive them from the implicit certificates and the certificate authority public keys. These schemes usually result in more compact certificates, since signatures do not need to be transmitted together with the users' public keys. In this thesis, implicit certificates are considered in the generation of vehicular certificates in established vehicular PKIs.

2.6 Summary

In this chapter, we explored the cryptographic background used in this document, motivated by the security services required for a given application. The given mathematical concepts are essential for asymmetric cryptographic protocols, with the choice of elliptic curve cryptography for its smaller parameters. Among the protocols applied in the proposed solution, we introduced hash functions for providing data integrity and authentication (when constructing a MAC), and a possible attack when a large

number of hashes is distributed. Encryption schemes are presented in both symmetric and asymmetric configurations and are used to provide confidentiality of the transmitted data, with or without authentication. Finally, digital signatures are employed for authentication and non-repudiation, while it still requires a Public Key Infrastructure (PKI) to guarantee ownership of published keys.

3 RELATED VEHICULAR PUBLIC KEY INFRASTRUCTURES

In this chapter, we review four of the most prominent Vehicular Public Key Infrastructures (VPKI) from the literature. First, we detail the inner workings of the Security Credential Management System (SCMS), which presents an efficient mechanism for issuing and linking pseudonym certificates, and is also the base for our proposed improvement to the revocation procedure. Then, we briefly describe its main competitor, the Cooperative-Intelligent Transport System (C-ITS), whose certificate issuance follows a more traditional approach. Finally, we present two other solutions based on activation codes, the Issue First Activate Later (IFAL) and the Binary-hash-tree-based Certificate Access Management (BCAM), which we compare with our proposal in terms of efficiency and privacy.

3.1 Security Credential Management System (SCMS)

The Security Credential Management System (SCMS) (WHYTE et al., 2013) (CAMP, 2016) (BRECHT et al., 2018) provides efficient mechanisms for the provisioning and revocation of pseudonym certificates in V2X communications. Combined, these mechanisms enable revocable privacy while ensuring that no single entity in the system can track honest vehicles. As a result, it addresses the needs of the "honest-but-curious" security model (KHODAEI; PAPADIMITRATOS, 2015), in which the system's entities are expected to follow the correct protocols, but may engage in passive attacks such as trying to infer sensitive information from the exchanged data (e.g.,

aiming to track vehicles). Actually, it also takes into account attacks in a security model slightly more powerful than the "honest-but-curious" approach, which we call "dishonest-if-allowed": the system's entities may engage in active attacks, subverting the protocols' execution, but only if (1) this would bring them some advantage (e.g., the ability to track vehicles) and (2) if such misbehavior can go undetected.

After SCMS was published and became one of the leading candidate designs for protecting V2X security in the US, different improvements to its design have been proposed (KUMAR; PETIT; WHYTE, 2017) (SIMPLICIO et al., 2018c) (SIMPLICIO et al., 2018b). Since the proposal hereby presented builds upon SCMS and can also take advantage of the aforementioned improvements, in what follows, we describe SCMS in some detail, giving an overview of the state of the art and the limitations targeted by our solution.

3.1.1 Overview

In SCMS, each vehicle receives two types of certificates: one enrollment certificate, which identifies authorized devices and is expected to have a long lifespan (e.g., years) before expiring; and multiple pseudonym certificates, each having a short expiration time (e.g., a few days), so only $\sigma \geqslant 1$ certificates of this type are valid simultaneously. For protecting their privacy, vehicles may alternate the pseudonym certificates employed when sending different messages, thus avoiding tracking by nearby vehicles or by roadside units. In practice, however, it is important that σ remains small to limit the effects of "Sybil-like" attacks (DOUCEUR, 2002), in which one vehicle poses as a platoon by sending multiple messages signed with different pseudonym certificates. Otherwise, a vehicle abusing its pseudonymity in this manner could, for example, receive preferential treatment from traffic lights programmed to give higher priority to congested roads (MOALLA et al., 2012).

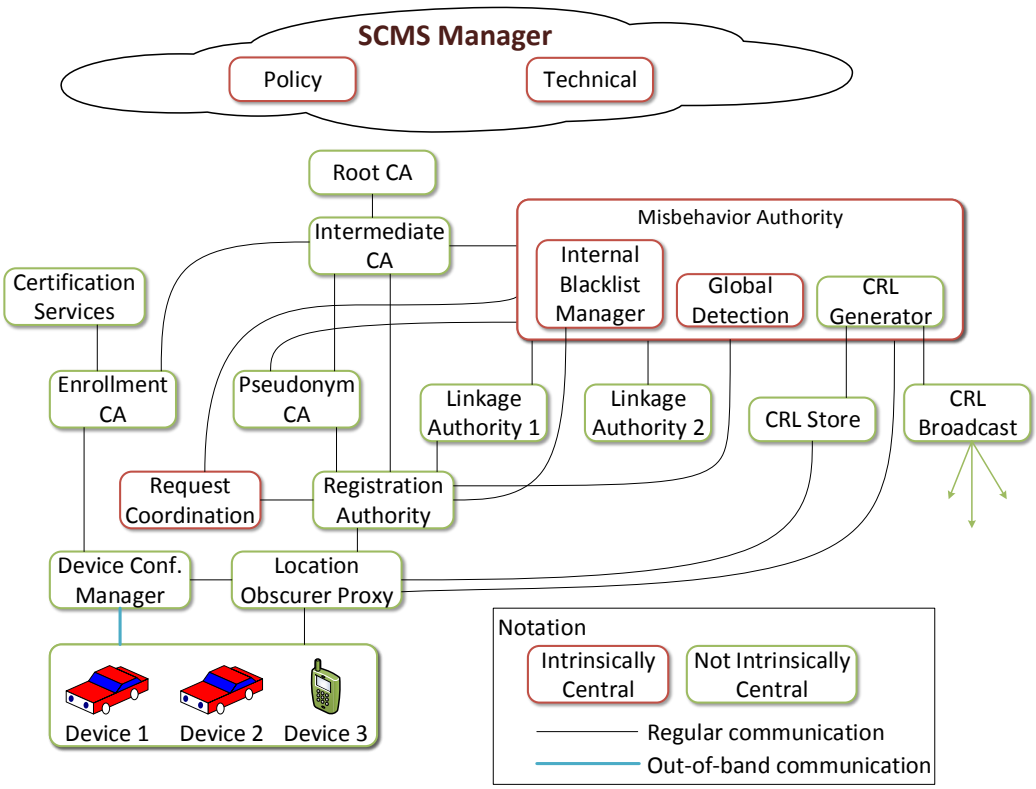
As previously mentioned, SCMS was designed to distribute multiple pseudonym

certificates to vehicles efficiently while providing mechanisms for easily revoking and linking them to their owners in case of misbehavior. For this purpose, SCMS relies basically on the following entities (see Figure 2 for a complete architecture, and (WHYTE et al., 2013) for the description of all of its elements):

- Pseudonym Certificate Authority (PCA): responsible for issuing pseudonym certificates to devices.
- Registration Authority (RA): receives and validates requests for batches of pseudonym certificates from devices identified by their enrollment certificates. Those requests are individually forwarded to the PCA, so requests associated to different devices are shuffled together so the PCA cannot link a set of requests to the same device.
- Linkage Authority (LA): generates random-like bitstrings that are added to certificates so they can be efficiently revoked (namely, multiple certificates belonging to the same device can be linked together by adding a small amount of information to certificate revocation lists – CRLs). SCMS uses two LAs, even though its architecture supports additional LAs.
- Misbehavior Authority (MA): identifies misbehavior by devices and, whenever necessary, revokes them by placing their certificate identifiers into a CRL.

These entities play different roles in the two main procedures provided by SCMS: the butterfly key expansion, which allows pseudonym certificates to be issued; and key linkage, which allows the efficient revocation of malicious vehicles. Both are described in the following sections.

Figure 2: SCMS overview.



Source: Whyte et al. (2013)

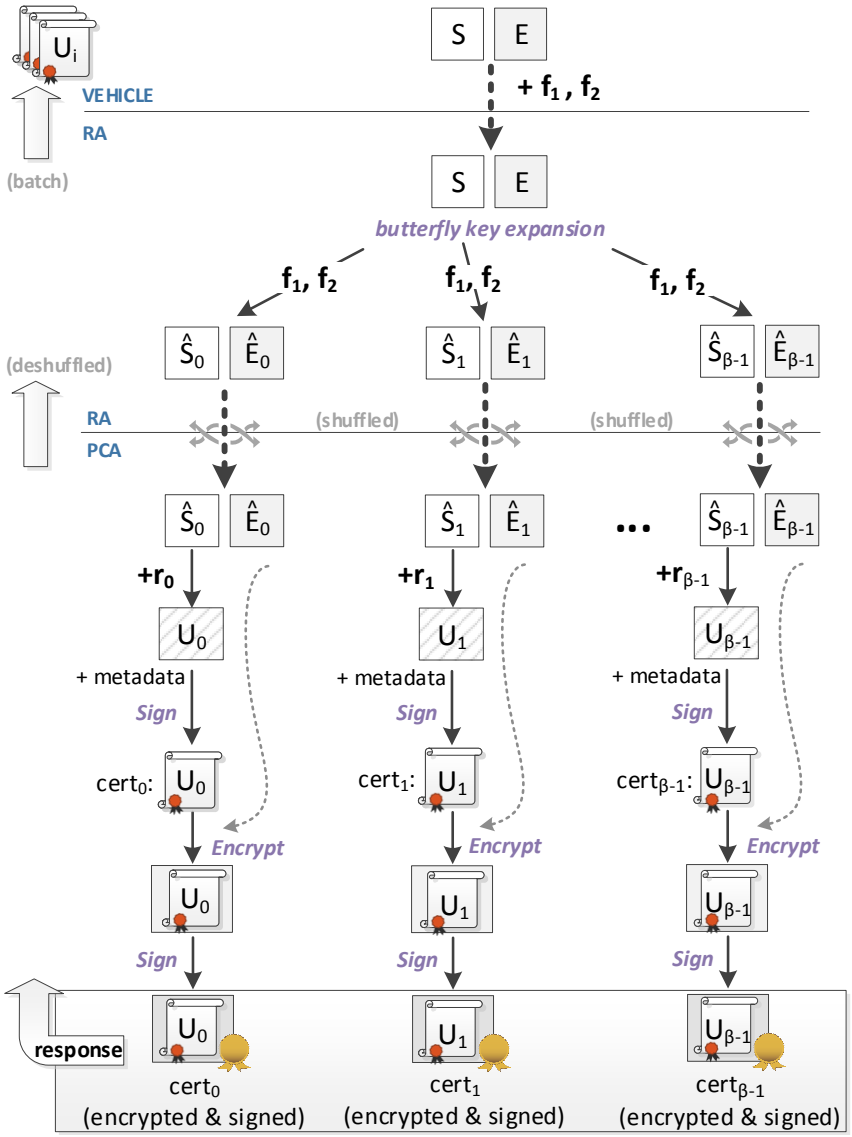
3.1.2 The Butterfly Key Expansion

The butterfly key expansion enables vehicles to obtain arbitrarily large sets of (short-lived) pseudonym certificates through a single, small-sized request message from vehicles. Figure 3 illustrates this process, whereas Table 3 lists the corresponding operations, explained in what follows.

The vehicle starts by picking two random private *caterpillar* keys, s and e, used respectively for signing messages and decrypting the batch of certificates. From them, the corresponding public caterpillar keys $S = s \cdot G$ and $E = e \cdot G$ are computed. It also picks two random seeds for initializing pseudorandom functions f_1 and f_2, used for deriving the original keys. The quadruple (S, f_1, E, f_2) is then sent to the Registration Authority (RA).

The RA then generates the batch of β public *cocoon* signature keys $\hat{S}_i = S + f_1(i) \cdot$

Figure 3: SCMS's butterfly key expansion and pseudonym certificate generation.



Source: the author.

G, where $0 \leqslant i < \beta$. Analogously, the RA uses E for generating β public cocoon encryption keys $\hat{E}_i = E + f_2(i) \cdot G$. Pairs of public cocoon keys $(\hat{S}_i, \hat{E}_i)$ from different vehicles are then shuffled together so that any two cocoon keys cannot be linked to the same vehicle. Then, they are sent individually to the Pseudonym Certificate Authority (PCA) for the generation of the corresponding pseudonym certificates.

The subsequent procedure followed by the PCA when processing $(\hat{S}_i, \hat{E}_i)$ depends on whether explicit or implicit certificates (CERTICOM, 2013) are employed. For

Table 3: The butterfly key expansion process for issuing pseudonym certificates.

	Vehicle	→	RA	→	PCA	-RA→	Vehicle
SCMS (explicit)	s, $S = s \cdot G$	S, f_1	$\hat{S}_i = S + f_1(i) \cdot G$	$\hat{S}_i$,	$\mathrm{U}_i = \hat{S}_i + r_i \cdot G$ $sig_i = Sign(u; \{\mathrm{U}_i, \texttt{meta}\})$ $cert_i = \{\mathrm{U}_i, \texttt{meta}, sig_i\}$ $pkg = Enc\left(\hat{E}_i; \{cert_i, r_i\}\right)$ $res = \{pkg, Sign(u; pkg)\}$	res	$\hat{e}_i = e + f_2(i)$ $Verif(\mathcal{U}; res)$ $\{cert_i, r_i\} = Dec(\hat{e}_i; pkg)$ $Verif(\mathcal{U}; cert_i)$ $\mathrm{u}_i = s + f_1(i) + r_i$ $\mathrm{u}_i \cdot G \stackrel{?}{=} \mathrm{U}_i$
SCMS (implicit)	e, $E = e \cdot G$	E, f_2	$\hat{E}_i = E + f_2(i) \cdot G$ $(0 \leqslant i < \beta)$	$\hat{E}_i$	$V_i = \hat{S}_i + r_i \cdot G$ $cert_i = \{V_i, \texttt{meta}\}$ $sig_i = Hash(cert_i) \cdot r_i + u$ $pkg = Enc\left(\hat{E}_i; \{cert_i, sig_i\}\right)$ $res = \{pkg, Sign(u; pkg)\}$		$\hat{e}_i = e + f_2(i)$ $Verif(\mathcal{U}; res)$ $\{cert_i, sig_i\} = Dec(\hat{e}_i; pkg)$ $h_i = Hash(cert_i)$ $\mathrm{u}_i = h_i \cdot (s + f_1(i)) + sig_i$ $\mathrm{U}_i = \mathrm{u}_i \cdot G \stackrel{?}{=} h_i \cdot V_i + \mathcal{U}$

Source: the author

explicit certificates, the PCA picks a random r_i and computes the vehicle's public signature key as $\mathrm{U}_i = \hat{S}_i + r_i \cdot G$. The PCA then digitally signs this public key together with any required metadata `meta` (e.g., that key's validity period), obtaining the signature sig_i, so the resulting explicit pseudonym certificate $cert_i$ is defined by the triple $(\mathrm{U}_i, \texttt{meta}, sig_i)$. For implicit certificates, this process is slightly different: the PCA starts by computing a credential $V_i = \hat{S}_i + r_i \cdot G$, once again for a random r_i, so the implicit certificate $cert_i$ is defined as the pair $(V_i, \texttt{meta})$. The PCA then signs this certificate to obtain $sig_i = h_i \cdot r_i + u$, where $h_i = Hash(cert_i)$ and u is the PCA private key. Whichever the certification model adopted, the resulting certificate and its companion data (namely, r_i for explicit certificates, and sig_i for implicit ones) are encrypted with $\hat{E}_i$, which can only be decrypted by the vehicle. To avoid Man-in-the-Middle (MitM) attacks by the RA, this encrypted package is also signed with the PCA's own private key u. The PCA's response is then sent to the RA, which gathers and de-shuffles them before relaying the corresponding certificates to the requesting vehicle.

Finally, the vehicle verifies the PCA's signature on the encrypted certificates, uses the private cocoon encryption key $\hat{e}_i = e + f_2(i)$ to decrypt the PCA's response, and verifies the validity of the pseudonym certificate thereby enclosed. For explicit certificates, this verification consists in checking that the certificate's signature is valid and

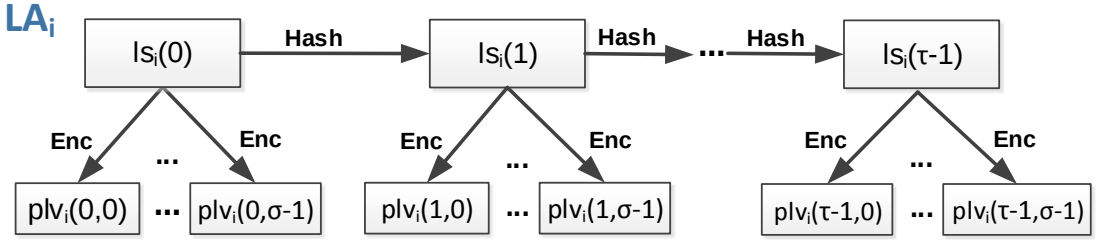
that the public key U_i was indeed derived from s and r_i, i.e. that $U_i = (s + f_1(i) + r_i) \cdot G$; in that case, the vehicle's private key is set to $u_i = s + f_1(i) + r_i$. When the certificates are implicit, the vehicle simply follows the regular implicit verification protocol described in (CERTICOM, 2013): it computes its own private key as $u_i = h_i \cdot (s + f_1(i)) + sig_i$, sets the corresponding public key to $U_i = u_i \cdot G$, and verifies that $U_i = h_i \cdot V_i + \mathcal{U}$, where $\mathcal{U}$ is the PCA's public signature key. These operations are also executed when other vehicles verify this certificate.

Independently of the type of certificate adopted, the vehicles' privacy is protected in this process as long as there is no collusion between RA and PCA. Specifically, the shuffling of public cocoon keys performed by the RA prevents the PCA from learning whether or not a group of keys received belongs to the same device. In turn, the unlinkability of public keys towards the RA is obtained because the latter does not learn the value of $cert_i$ in the PCA's encrypted response.

3.1.3 Key Linkage

To avoid large certificate revocation lists (CRLs), revocation in SCMS is done so that many certificates from the same user can be linked together by inserting only a small amount of information into a CRL. For this purpose, each pseudonym certificate generated by the PCA includes a linkage value `lv`, computed by XORing two (or more) pre-linkage values, $\texttt{plv}_1$ and $\texttt{plv}_2$, provided by different Linkage Authorities (LA). The generation of $\texttt{plv}_i$ by LA_i is done upon request by the RA, as follows (see Figure 4).

Figure 4: SCMS's key linkage process: generation, by LA_i, of pre-linkage values employed for certificate revocation.



Source: the author

First, LA_i picks a random, 128-bit linkage seed $\texttt{ls}_i(0)$. Next, if the RA's request covers τ certificate time periods, LA_i iteratively computes a τ-long hash chain (LAMPORT, 1981) $\texttt{ls}_i(t) = \mathit{Hash}(\mathit{la\_id}_i \,||\, \texttt{ls}_i(t-1))$, where $\mathit{la\_id}_i$ is LA_i's identity string and $1 \leqslant t < \tau$. Each $\texttt{ls}_i(t)$ is then used in the computation of σ pre-linkage values $\texttt{plv}_i(t, c) = \mathit{Enc}\,(\texttt{ls}_i(t); \mathit{la\_id}_i \,||\, c)$, for $0 \leqslant c < \sigma$, for σ certificates valid in each time period. Finally, every $\texttt{plv}_i(t, c)$ is truncated to a suitable length, individually encrypted and authenticated using a key shared between the PCA and LA_i, and then sent to the RA. The RA simply includes this encrypted information, together with the corresponding cocoon keys, in the requests sent to the PCA. As a result, the latter can compute the linkage value to be included in the c-th certificate valid in time period t as $\texttt{lv}(t, c) = \texttt{plv}_1(t, c) \oplus \texttt{plv}_2(t, c)$.

Whenever the Misbehavior Authority (MA) identifies a malicious vehicle, its non-expired certificates can be revoked altogether. It is accomplished via the collaboration among PCA, RA, and LAs. Namely, the PCA can associate the $\texttt{lv}$ informed by the MA to the original pseudonym certificate request received from the RA. The PCA then provides this information, together with the corresponding pre-linkage values $\texttt{plv}_i(t, c)$, to the RA. In turn, the RAcan (1) identify the vehicle/driver behind that certificate request, placing the corresponding enrollment certificate in a blacklist for preventing its owner from obtaining new pseudonym certificates; and (2) ask each LA_i to identify the linkage seed $\texttt{ls}_i(0)$ from which $\texttt{plv}_i(t, c)$ was computed. Finally, each LA_i provides RA with $\texttt{ls}_i(t_s)$, where t_s is the time period from which the revocation starts being valid (e.g., the time period when the misbehavior first occurred, or the current one). The set of $\texttt{ls}_i(t_s)$ received from the LAs can then be placed in a CRL to be distributed throughout the system, allowing any entity to compute $\texttt{lv}(t, c)$ for time periods $t \geqslant t_s$, linking the corresponding certificates to a single CRL entry. Consequently, the misbehaving vehicle's *future* certificates are revoked and can be linked together; *past* certificates remain protected, though, preserving the vehicle's privacy prior to the detection of

the malicious activity. The re-enrollment of revoked vehicles, if allowed, would then require a new enrollment certificate to be issued, as well as a new execution of the pseudonym certificate provisioning protocol.

In terms of computational costs, this revocation process is such that each revoked device results in 2 pre-linkage values added to the CRL. Hence, the CRL grows linearly with the number of revoked vehicles, not with the number of revoked certificates. The main drawback of this gain in size is that checking whether a given certificate is in the CRL requires the verification of *every* CRL entry against that certificate's linkage value. More precisely, for each CRL entry published at time period t_s, the verification of whether it covers a given certificate involves basically the computation of two components:

a) $\texttt{ls}_i(t_c)$: it takes $2 \cdot (t_c - t_s)$ hashes to compute $\texttt{ls}_i(t_c)$ from $\texttt{ls}_i(t_s)$, where $i = \{1, 2\}$ and t_c is the time period when the verification is performed. This cost may be reduced by means of pre-computation, i.e., if the vehicles always keep the updated version of the linkage seeds, $\texttt{ls}_i(t_c)$, besides the original ones provided in the CRL. Nonetheless, to cope with the lack of a system-wide time synchronization (VERHEUL, 2016), vehicles may actually need to keep a slightly older linkage seed in memory; for example, by keeping $\texttt{ls}_i(t_c - \epsilon)$ for a small ϵ, it is possible to compute $\texttt{ls}_i(t_c)$ with only ϵ hashes.

b) $\texttt{plv}_i(t_c, c)$: since the certificate under analysis may be any out of σ that are valid in the current time period, it takes up to σ encryptions to compute each $\texttt{plv}_i(t_c, c)$ from $\texttt{ls}_i(t_c)$. With enough memory, the latency of this process can be reduced via the pre-computation of a look-up table with all possible σ entries for each $\texttt{ls}_i(t_c)$ in the CRL.

3.2 Cooperative-Intelligent Transport System (C-ITS)

The Cooperative Intelligent Transport System (C-ITS) is a set of technical reports and standards published by the European Telecommunications Standards Institute (ETSI) that provides definitions for device specifications, certificate management, and communication protocols to guarantee secure V2X communication (ETSI, 2021) (ETSI, 2017a) (ETSI, 2010). Although it is considered the main competitor against SCMS, it presents several similarities in handling short-lived pseudonym certificates and the role division among the authorities for privacy. Their main difference, however, results from the strategy used in the issuance of C-ITS pseudonym certificates: instead of using the butterfly key expansion, they rely on generating individual pseudonym certificates on demand. This single difference modifies how the authorities have to handle privacy, how pseudonym certificates are linked to vehicles, and how the revocation status is verified.

In what follows, we describe C-ITS focusing on its differences regarding SCMS. Given their similarities, many terms and expressions are analogous between them. So, to maintain clarity in the comparison, we present the expressions proposed by C-ITS but will adopt the ones defined by SCMS in the comparison thereafter.

3.2.1 Overview

Similar to SCMS, each vehicle requests a long-term enrollment certificate, which can be used to retrieve several short-term pseudonym certificates (called *Activation Tickets*). While the enrollment certificate uniquely identifies the vehicle, each vehicle has access to $\sigma \geqslant 1$ valid pseudonym certificates simultaneously, which can be used alternately in a route to provide privacy. On the other hand, differently from SCMS, the pseudonym certificates are only issued on demand. Before a given time period, the vehicle requests pseudonym certificates which are only valid until the time period

ends. This limitation actually has an advantage: the system authorities do not need to maintain a CRL for the pseudonyms. Since C-ITS pseudonym certificates are only valid for a short period, it also proposes that the revocation status of pseudonyms is not necessary. However, if the pseudonyms are misused, the authorities respond by revoking the vehicle's enrollment certificate, which makes it unable to be authenticated in the system and thus cannot obtain new pseudonyms for the next time periods.

Given the simpler solution to obtain pseudonym certificates, C-ITS relies upon only two basic entities to generate device certificates (the complete architecture of C-ITS can be seen in (ETSI, 2010)):

- Enrollment Authority: similar to SCMS's Registration Authority (RA), it is responsible for generating the vehicle's enrollment certificate and authorizing requests to issue pseudonym certificates.
- Authorization Authority: similar to SCMS's Pseudonym Certificate Authority (PCA), it is responsible for issuing pseudonym certificates if the vehicles' credentials are valid.

The lack of an entity equivalent to SCMS's Linkage Authorities (LA) limits how C-ITS can link pseudonyms of dishonest vehicles: instead of using the linkage values from the certificates, the PCA alone can identify the owner of the pseudonym. While this feature reduces the vehicles' privacy towards the system authorities, the vehicles gain in efficiency as they do not need to verify the validity of pseudonyms. In order to better analyze this trade-off, we describe the pseudonym issuance method in the following section.

3.2.2 Pseudonym Issuance and Linkage

In order to obtain pseudonym certificates for a given time period, the vehicle (called *ITS Station*) must request them individually from the PCA. For each

pseudonym, it generates a random pseudonym private key u_i and computes the equivalent public key $U_i = u_i \cdot G$. The public key is then signed using the vehicle enrollment certificate key $sig(U_i)$ and is encrypted using the RA public key. The tuple $(U_i, Enc\,(pk_{RA}; (sig(U_i)))$ is sent as a request to the PCA.

Upon reception, the PCA relays the encrypted part of the request to the RA, so it can authenticate the user requesting new pseudonyms. The RA verifies the signature to the public key, asserting that this request originates from the vehicle and not from a rogue PCA. If the signature is valid, the RA authorizes the PCA to sign the pseudonym key $sig_i = Sign\,(sk_{PCA}; U_i)$, and return it to the vehicle. The PCA also stores that the pseudonym key U_i belongs to the vehicle that requested it.

In order to link pseudonyms to the vehicle's identity, the PCA maintains a registry of every pseudonym certificate it has authorized. So, whenever a given pseudonym must be revoked, the PCA simply looks to which vehicle it belongs and requests the RA to revoke its enrollment certificate. Whenever the revoked vehicle requests new pseudonym certificates, the RA is able to gauge the revocation status and deny the PCA to sign the certificate.

3.3 Issue First Activate Later (IFAL)

The Issue First Activate Later (IFAL) (VERHEUL; HICKS; GARCIA, 2019) (VERHEUL, 2016) was proposed as an improvement to C-ITS for issuing pseudonym certificates more efficiently. It follows an "honest-but-curious" threat model, so attackers can only listen to correctly executed messages but cannot modify their contents to gain advantage. This is expected if we consider that C-ITS follows the same approach by design, allowing the PCA to link pseudonym certificates to the vehicles in order to be able to revoke them when necessary. However, it requires an additional assumption that the vehicle contains a trusted element (TE) capable of signing messages, but

considered that it cannot be compromised.

Similar to the previous VPKIs, each vehicle is registered with a single enrollment certificate, which uniquely identifies it to the system authorities. However, instead of following C-ITS issuance process of creating each certificate individually, IFAL allows the vehicles to retrieve short-lived pseudonym certificates for several time periods (or *epochs*) up to the vehicle's lifespan. Although SCMS provides the same feature, IFAL's approach does not directly disclose these certificates to the vehicles, but limits their access until their activation. For that, the private keys from each certificate can only be derived after receiving the activation code from the system authorities. This way, not only will the vehicle have access to very few valid certificates in each epoch, but a revocation will restrict it from using certificates for the following time periods.

Since IFAL is built upon the C-ITS architecture, it also relies on the same authorities to generate certificates. However, in order to use activation codes, the flow of messages and the authorities' responsibilities were adapted. The RA is not only responsible for generating the vehicle's enrollment certificate but is also required to distribute the activation codes to non-revoked vehicles. And the PCA becomes responsible for issuing pseudonym certificates covering the vehicle's entire lifespan, and releasing all activation codes to the RA each time period, so that the latter can discard the revoked ones.

As with C-ITS, there is no LA to link pseudonyms the vehicles' identity. The linkage is executed by the PCA, which can identify the vehicle's identity from the request for pseudonym certificates, and pseudonym certificates do not need to be revoked due to their short validity. For comparison with this proposal, we now present the details of the pseudonym issuance procedure, distribution of activation codes, and the pseudonym certificate usage in the following sections.

3.3.1 Pseudonym Issuance

After obtaining the enrollment certificate from the RA, the vehicle is enabled to request pseudonym certificates. The vehicle's TE creates a random pseudonym private key u_{TE} and computes the equivalent public key $\mathrm{U}_{TE} = \mathrm{u}_{TE} \cdot G$. The public key U_{TE} is shared with the vehicle, which creates a pseudonym request to the PCA with the public key and the enrollment credentials.

After verifying the vehicle credentials, the PCA uses U_{TE} as a random seed which is used to derive the other pseudonym keys U_i. For that, the PCA creates a random batch key $\mathtt{kb}_\tau$ for each epoch τ, and uses a pseudorandom function f_1 in the creation of the pseudonym public keys: $\mathrm{U}_i = f_1(\mathtt{kb}_\tau, i) \cdot \mathrm{U}_{TE}$. These keys are then signed using a variant of the deterministic ECDSA signature scheme (PORNIN, 2013) and returned to the vehicle as certificates. The batch keys $\mathtt{kb}_\tau$ are stored by the PCA as the vehicles' activation codes.

3.3.2 Activation Code Distribution and Certificate Usage

The activation of vehicles happens periodically, just before a new epoch begins. The PCA will deliver the activation codes for every vehicle to the RA, from which it discards the revoked ones. Then, each vehicle can request the RA for its own batch key $\mathtt{kb}_\tau$ for the following time-period τ. If the activation code was not discarded, the RA returns it to the vehicle.

With the batch key, the vehicle is activated to create signatures with the pseudonym certificates. For the i-th certificate in the batch, the vehicle computes the epoch key $k_{\tau,i} = f_1(\mathtt{kb}_\tau, i)$, using the same pseudorandom function f_1 as the PCA. By using a process similar to Chaum's blind signature (CHAUM, 1983), the vehicle first transforms the message str using the epoch key into $str' = T_1(str, k_{\tau,i})$ and send it to the TE. The TE signs the message with its base key, resulting in a signature $sig' = Sign\,(\mathrm{u}_{TE}; str')$.

Then the vehicle transforms the signature into $sig = T_2(sig', k_{\tau,i})$, which can be relayed to other vehicles. The details of the transformations (T_1, T_2) can be found in (VERHEUL; HICKS; GARCIA, 2019, Alg. 3 and 4).

3.4 Binary-hash-tree-based Certificate Access Management (BCAM)

The Binary-hash-tree-based Certificate Access Management (BCAM) (KUMAR; PETIT; WHYTE, 2017) was proposed as an improvement to SCMS to reduce the costs of verifying the vehicles' revocation status, and to allow wrongly revoked vehicles to be reinstated in the system without a fresh registration. They proposed using *activation codes* to encrypt pseudonym certificates during issuance so that only non-revoked vehicles would be able to decrypt (i.e., activate) them. To achieve these goals, BCAM adapts the use of activation codes to be obtained from a binary tree, where vehicles are represented as leaves, and all nodes are derived from the root. Therefore, it eliminates the need for bidirectional connectivity when requesting activation codes, as it can broadcast inner nodes capable of activating several vehicles simultaneously.

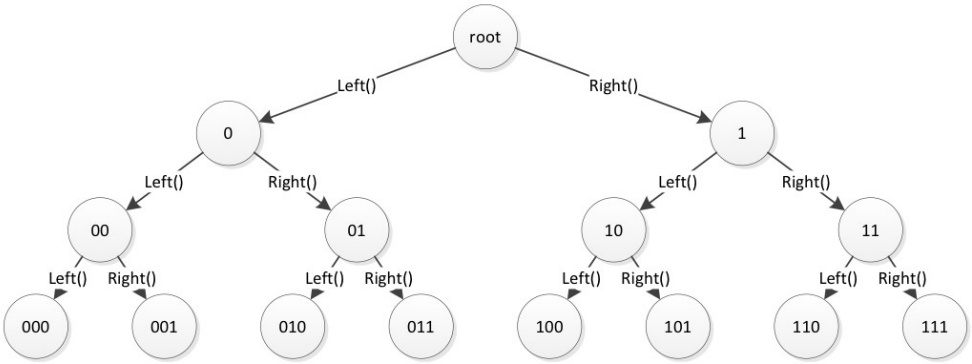
The generation of pseudonym certificates is still much similar to SCMS, as it uses the butterfly key expansion to derive certificates from a seed and then uses linkage values to later link the pseudonyms of a revoked vehicle. However, the activation codes are managed by a new authority called the Certificate Access Manager (CAM). The responsibilities of the CAM are threefold: (1) to create activation trees for each time period; (2) to encrypt batches of pseudonym certificates from the same vehicle for a given time period; and (3) to manage which tree nodes should be delivered to activate the certificates of only non-revoked vehicles. The CAM is not considered a central entity to the SCMS architecture, and such as LAs, several CAMs can be used to create the activation codes.

In the following sections, we detail the functionalities added by BCAM.

3.4.1 Creation of Activation Trees

For each CAM responsible for creating activation codes, a complete balanced binary tree is constructed for a given time-period τ. Each tree leaf represents a vehicle with identification VID, and the tree nodes are derived from the root, with each 0 leading to the child on the left and each 1 to the child on the right, as presented in Figure 5. This way, to span enough leaves to cover all vehicles, the tree must have depth $D = |VID|$. Which, by following BCAM recommendation (KUMAR; PETIT; WHYTE, 2017, Sec. 4.1.1), results of VID with length of 40 bits (thus, trees with depth $D = 40$), corresponding to more then 1 trillion vehicles.

Figure 5: Construction of BCAM binary tree



Source: adapted from Kumar, Petit and Whyte (2017, Fig. 3)

The activation tree is constructed iteractively using a collision-resistance hash function, starting from a random $\texttt{root}_\tau = \texttt{node}_\tau(0, 0)$. When the entire tree is created, the leaf nodes (called *device-specific values* – DSV) are used to generate the BCAM batch key to encrypt/decrypt certificates. To generate the tree, from any node $\texttt{node}_\tau(depth, count)$, its children nodes are computed as the following:

- Left child: $\texttt{node}_\tau(depth + 1, 2 \cdot count) = Hash\left(\texttt{node}_\tau(depth, count) \,||\, 0^l\right)$
- Right child: $\texttt{node}_\tau(depth + 1, 2 \cdot count + 1) = Hash\left(\texttt{node}_\tau(depth, count) \,||\, 1^l\right)$

The node length and the padding length l is chosen in BCAM by the underlying

hash function used in its construction. To provide 128 bits of security and nodes of 256 bits, they elected the hash function SHA-256. For efficiency, its compression function can be called only once if the padding length is at most 196 bits due to its 512-bit input and 64-bit length indicator.

3.4.2 Issuance of Pseudonym Certificates

The issuance of pseudonym certificates follows a similar path to the butterfly key expansion in SCMS (see Section 3.1.2). First, the vehicle generates two caterpillar key pairs (s, S, e, E), one for signature and one for encryption, and initialize the pseudorandom functions (f_1, f_2) used to derive the keys, which are sent to the RA to obtain new certificates. Then, the RA expands the keys producing all the cocoon keys $(\hat{S}_i, \hat{E}_i)$ for the vehicle's lifetime and stores them. After receiving requests from several vehicles, the RA shuffles the cocoon keys from the different vehicles before sending them to the PCA, so the cocoon keys cannot be linked to a single vehicle. The PCA transforms each received signature key into the respective butterfly key (U_i), digitally signs and encrypts them with the vehicle's expanded encryption key. These certificates are returned to the RA, which unshuffles the encrypted butterfly keys, so it has batches of (encrypted) certificates from the same vehicle.

Only after this point, the activation codes are used. After receiving the signed butterfly keys, the RA assembles for a given time period τ a set of certificates that belongs to the same vehicle. With activation tree already created, the CAM computes the DSV $\mathtt{dsv}_{\mathrm{VID},\tau}$ regarding the requesting vehicle VID from the tree root. From this value, it uses a KDF to create the BCAM batch key $\mathtt{kb} = \mathit{KDF}\,(\mathtt{dsv}_{\mathrm{VID},\tau})$. Then, each certificate is re-encrypted using a symmetric key encryption scheme, with DSV as its key, resulting in the response $res = \{\mathit{Enc}\,(\mathtt{kb}; \mathrm{U}_i)\}$, for all U_i in the time-period τ. The re-encrypted certificates res are then returned to the RA, which can repeat this process for different vehicles and different time periods. After all certificates from a given

vehicle have been re-encrypted, they are returned to their owners.

Without access to the activation codes, the last stage of the butterfly key expansion is delayed since the vehicles cannot decrypt the received certificates. Only after the vehicles are provided with their activation codes, right before the time period of the next batch of certificates, can they derive their DSV for removing the CAM layer of encryption. Then, the vehicle can complete the issuance process by removing the PCA layer of encryption on the certificates. The signature private key u is finally derived from the private caterpillar key, and the PCA signature on the pseudonym certificate is verified to guarantee the correctness of the execution.

3.4.3 Distribution of Activation Codes

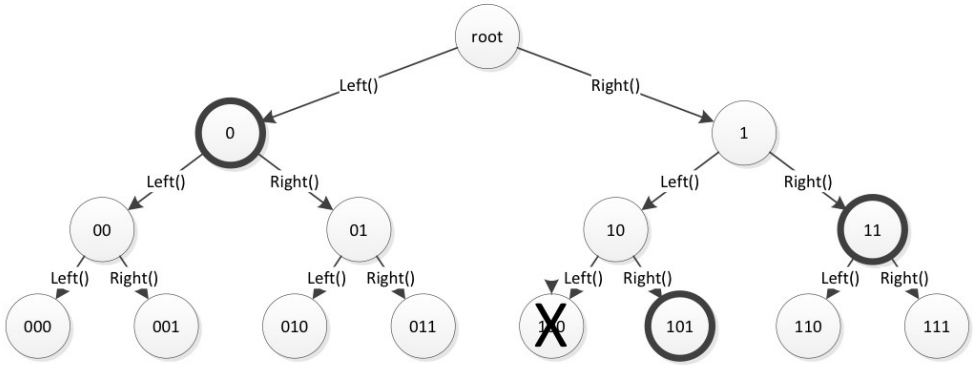
When a time period is close to begin, the CAM is supposed to send the activation codes to all non-revoked vehicles so they can access their pseudonym certificates. BCAM proposes that the activation codes be transmitted using a broadcast channel to all vehicles so that bidirectional connectivity is not required. If no vehicles are revoked, the CAM only needs to broadcast the root of the tree, and all vehicles will be able to derive their DSV. However, to stop revoked vehicles from decrypting their certificates, not all nodes of the activation tree should be available. Even if the revoked vehicle does not receive their activation code directly, collusion among non-revoked and revoked vehicles would allow the latter to activate its certificates if its DSV could be derived from the transmitted nodes. Thus, to avoid the illegal activation of certificates, the activation tree must be "pruned" from all leaves representing these revoked vehicles.

In order to choose the nodes to be removed, the RA should maintain a list of the VIDs of all revoked vehicles. For each removed VID, the leaf it corresponds should to be removed, and every node in the path from the leaf to the root cannot be disclosed. And to activate all the remaining nodes in the tree, the CAM selects the sibling nodes

for each node in the path. By repeating the process to all revoked nodes, the CAM would return all the nodes that derive the DSV of only the valid vehicles.

As an example, using the tree of $depth = 3$ from Figure 6, when revoking the vehicle with VID = 100 (the node marked with an X), all the nodes from the `root` to the leaf should not be distributed. So, the CAM can transmit the nodes marked with bold border $(0, 11, 101)$, which are the siblings to the nodes removed.

Figure 6: Revocation of node with VID = 100 in the BCAM binary tree of $depth = 3$. Only the nodes with bold borders $(0, 11, 101)$ should be broadcast.



Source: adapted from Kumar, Petit and Whyte (2017, Fig. 3)

3.5 Summary

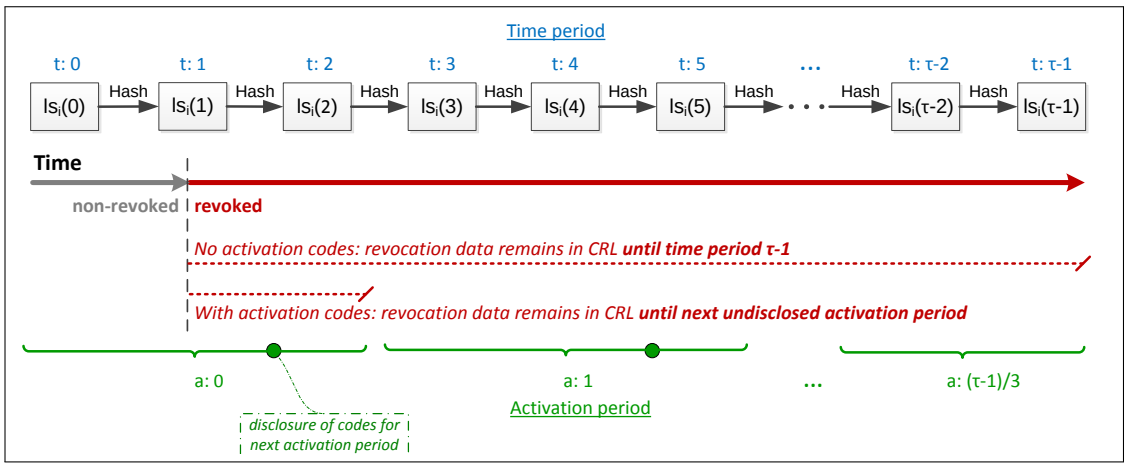
In this chapter, we presented the main VPKI schemes in the literature. We detailed the methods of the Security Credential Management System (SCMS), which is the basis for the improvements in this document, presenting its architecture and the process to efficiently issue and revoke pseudonym certificates. We briefly described its main competitor, the Cooperative Intelligent Transport System (C-ITS), which focuses on providing certificates on-demand so that it does not require the management of revoked pseudonyms. Then, we summarize the main methods of the Issue First Activate Later (IFAL), which improves upon the C-ITS to generate pseudonyms off-band, but only allowing them to be accessed after receiving activation codes from the system authorities. And lastly, we describe the Binary Hash Tree based Certificate Access

Management (BCAM), which improves the SCMS revocation of pseudonyms by requiring activation codes to decrypt the issued certificates, and removes the need for bidirectional connectivity to distribute the activation codes.

4 PROPOSAL: ACTIVATION CODES FOR PSEUDONYM CERTIFICATES (ACPC)

To avoid the growth of CRLs while preserving the performance gains associated with the butterfly key derivation, we build upon the concept of *activation codes*: bit-strings without which certificates previously acquired cannot be used (namely, in the proposed solution, they cannot be decrypted). Each activation code enables the usage of certificates corresponding to a certain *activation period*, which spans $\alpha \geqslant 1$ certificate time periods. This is illustrated in Figure 7, for activation periods covering $\alpha = 3$ time periods.

Figure 7: Activation codes and permanence of revocation data (e.g., linkage seeds) in CRLs.



Source: the author

Those codes are then periodically disclosed to non-revoked vehicles, which should

be done before the start of the corresponding time periods to allow a timely activation of their own certificates. Revoked vehicles are prevented from obtaining activation codes for their certificates, at least until this revocation status is removed. As a result, identifiers of revoked certificates that cannot be activated do not need to remain in CRLs, reducing the sizes of the latter. For example, certificates could be valid for 1 week, whereas the activation period could be set to 3 weeks and be disclosed 1 week before they are actually required. In this case, identifiers for certificates from revoked vehicles would have to remain in CRLs for at most 4 weeks: if the codes for the next activation period have not yet been disclosed, then the CRL needs to cover only the current activation period (at most 3 weeks); otherwise, the CRL needs to cover the next activation period (3 weeks) and the remainder of the current one (at most 1 week). After that, revoked devices do not receive new activation codes.

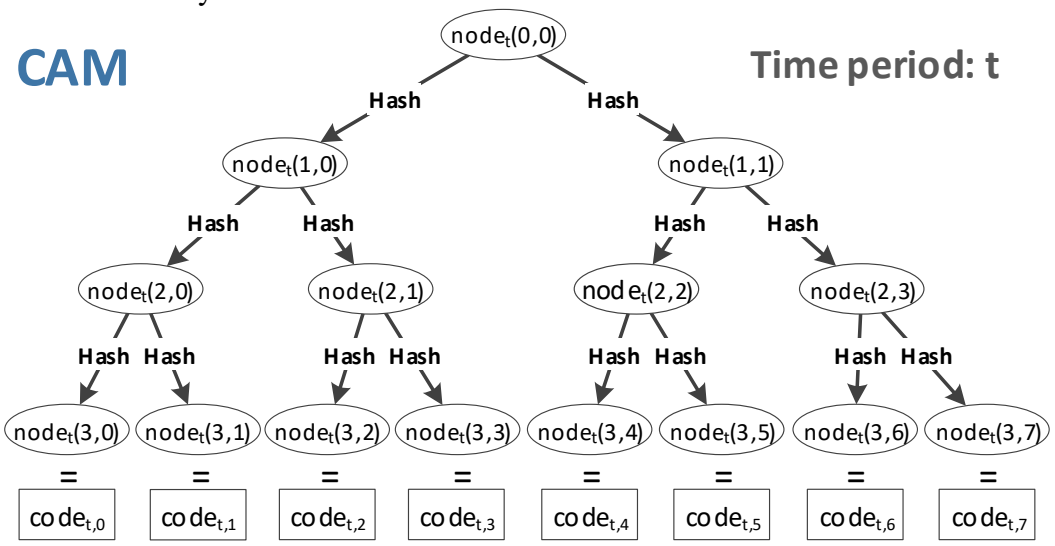
The proposed solution, named ACPC (Activation Codes for Pseudonym Certificates), is inspired by works such as IFAL (VERHEUL, 2016) and BCAM (KUMAR; PETIT; WHYTE, 2017). However, it innovates by addressing the main shortcomings of these solutions in terms of performance and privacy, as further discussed in Section 6.

4.1 Generating Activation Codes: Binary Hash Trees

Similarly to BCAM, we assume the existence of one or more Certificate Access Managers (CAMs), each having a different identifier *cam_id*. CAMs are entities responsible for creating and distributing activation codes. For this purpose, a CAM creates a binary hash tree $\texttt{tree}_t$ for each time period t, using a preimage resistant hash function, as illustrated in Figure 8. If the activation period spans $n + 1$ time periods, then $\texttt{tree}_t = \ldots = \texttt{tree}_{t+n}$.

The tree's nodes are denoted $\texttt{node}_t(depth, count)$, where $depth \geqslant 0$ and $0 \leqslant$

Figure 8: Activation tree generated by CAM. The activation codes correspond to the leaves of the binary hash tree.



Source: the author

$count \leqslant 2^{depth} - 1$ indicate the node's position inside the tree. The depth D of the tree must match the length of the vehicles' identifiers (VID), in bits. As a result, each leaf $\texttt{node}_t(depth, count)$ can be used to represent a single vehicle in the system: the one with VID = $count$. In (KUMAR; PETIT; WHYTE, 2017), for example, the suggested length of VID is 40 bits, which is enough to cover more than 1 trillion vehicles. For brevity of notation, we denote by $\texttt{code}_{t,\text{VID}}$ the leaf of $\texttt{tree}_t$ whose index corresponds to a given VID, i.e., $\texttt{code}_{t,\text{VID}} = \texttt{node}_t(|\text{VID}|, \text{VID})$.

The nodes of $\texttt{tree}_t$ are assumed to be k-bit long, yielding a k-bit security level (e.g., in modern deployments one would expect $k = 128$). The tree is built in the following manner. First, its root $\texttt{node}_t(0,0)$ is set to a (pseudo)random bitstring, unique for each activation period. Every other node is then computed from its parent node combined with a "security string" I, i.e., a different suffix for each node. More precisely, we have:

$$\texttt{node}_t(0,0) = \{0,1\}^k \qquad \text{(picked at random)}$$

$$\texttt{node}_t(depth, count) = Hash(\texttt{node}_t(depth-1, \lfloor count/2 \rfloor) \,||\, I)$$

where the security string I is defined as

$$I = (cam\_id \,||\, t \,||\, depth \,||\, count)$$

If the activation period spans multiple time periods, then t is set to the first time period covered by that activation period. This approach gives the system enough flexibility to increase or reduce the length of the activation periods without incurring the repetition of security strings. As further discussed in Appendix A, such non-repeatable security strings are useful to thwart birthday attacks analogous to those described in (BIHAM, 2002) (SIMPLICIO et al., 2018b).

Table 4 shows suggested lengths for the fields that compose those security strings in ACPC, leading to $|I| = 96$ bits. This composition is designed to support 40-bit long VIDs for 2^{16} time periods, which means more than 1200 years if the time periods are 1 week long. At the same time, adding security strings to this computation is unlikely to have any perceptible impact on processing times for building activation trees, as long as the hash function's input fits its block size. For example, SHA-256 operates on 512-bit blocks, appending at least 65 bits to its input message (a bit '1' for padding, and a 64-bit length indicator) (NIST, 2015); therefore, a single call to its underlying compression function is enough to process a 128-bit node value even when it is combined with a 319-bit or smaller security string.

4.2 Issuing Pseudonym Certificates with Activation Codes

Figure 9 illustrates ACPC's pseudonym certificate issuance, in which the binary hash tree generated by a CAM is integrated into SCMS's butterfly key expansion process. As shown in this figure, when a vehicle with a given VID requests a batch of pseudonym certificates, it still provides the quadruple (S, f_1, E, f_2) to the RA as de-

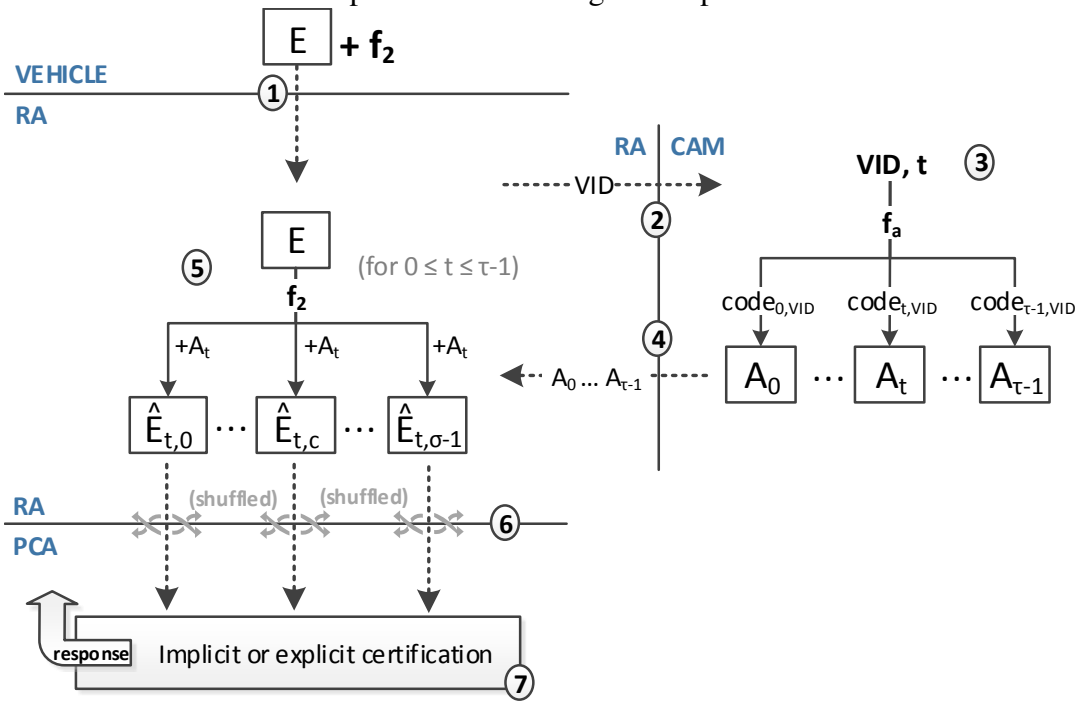
Table 4: Components of the security strings employed in ACPC's activation trees.

Field	Suggested length (bits)	Description
depth	8	Node's depth in tree, starting at 0. Mandatory: $\|depth\| \geqslant \lg(\|\text{VID}\|)$.
count	40	Node's index in the depth, starting at 0. Mandatory: $\|count\| \geqslant \|\text{VID}\|$.
t	16	Time period to which the tree is associated
cam_id	32	CAM's identifier

Source: the author

scribed in Section 3.1.2. It should then lead to the generation of $\beta = \tau \cdot \sigma$ certificates, providing a total of σ certificates are valid for each of the τ time periods covered in the batch, as usually done in SCMS.

Figure 9: Issuing pseudonym certificates with activation codes. For simplicity, we assume that each activation period covers a single time period.



Source: the author

Upon reception of the vehicle's request, the RA queries a CAM for the blinded activation value $A_t = f_a(\text{code}_{t,\text{VID}}, t, \text{VID}) \cdot G$. This operation uses the pseudorandom

function f_a, which could be instantiated, for example, using a NIST-approved construction (NIST, 2009) and $\mathsf{code}_{t,\mathrm{VID}}$ as secret seed. As a result, f_a's output is unpredictable because the activation tree's leaf $\mathsf{code}_{t,\mathrm{VID}}$ has not yet been disclosed by the CAM. In addition, f_a's output is blinded by the CAM via the multiplication by the elliptic curve generator G, so the actual unblinded activation value cannot be learned by the RA from the CAM's response. We note that, even though we assume for simplicity that a single CAM participates in the generation of certificate batches, in practice multiple CAMs could be contacted by the RA during this process, thus improving the system's resilience against the possibility of a CAM being compromised.

The RA then proceeds with the butterfly key expansion in a manner that is very similar to the process described in Section 3.1.2, except for one important difference: the blinded activation values provided by the CAM are added to the computation of the public cocoon encryption keys. More precisely, let $\hat{E}_{t,c}$ denote the c-th public cocoon encryption key that belongs to time period t. Each of those keys is now computed by the RA as $\hat{E}_{t,c} = E + A_t + f_2(t{\cdot}\sigma + c) \cdot G$, where $0 \leqslant c < \sigma$ and $0 \leqslant t < \tau$. As usual, the resulting cocoon keys are shuffled together with keys from other vehicles, before being sent to the PCA. By using f_2 in the computation of cocoon encryption keys, the RA ensures that they cannot be later correlated by the CAM or by the PCA, even for groups of keys computed using the same A_t. Therefore, this process preserves the unlinkability of pseudonym certificate requests, whether or not there is a collusion between CAM and PCA.

Finally, the PCA computes the vehicle's (implicit or explicit) pseudonym certificate, which is encrypted with $\hat{E}_{t,c}$ before being sent back to the RA. The RA, in turn, simply relays those certificates to the corresponding vehicle, without contacting the CAM once again. Since this procedure is identical to the one proposed in SCMS, the processing costs and bandwidth usage at the PCA remain unchanged. In addition, the underlying security properties discussed in Section 3.1.2 still apply, including the

protection against MitM attacks performed by the RA and the unlinkability among certificates unless PCA and RA collude.

The result of this process is that the encrypted certificates obtained by the vehicle can only be decrypted if the associated $\texttt{code}_{t,\text{VID}}$ is also obtained. After all, the decryption key corresponding to $\hat{E}_{t,c}$ is now computed as $\hat{e}_{t,c} = e + f_a(\texttt{code}_{t,\text{VID}}, t, \text{VID}) + f_2(t{\cdot}\sigma + c)$. Therefore, to keep a vehicle with identifier VID_r from activating its own certificates, it suffices to prevent it from obtaining $\texttt{code}_{t,\text{VID}_r}$. In that case, entries for that vehicles' certificates do not need to remain in CRLs for too long.

As a final remark, we note that the activation codes as hereby described can also be combined with the optimized, unified butterfly key expansion process proposed in (SIMPLICIO et al., 2018c). More precisely, this alternate design achieves better performance by replacing the pair of caterpillar keys (S, E) with a single key $X = x \cdot G$. Therefore, since the corresponding public cocoon keys $\widehat{X}$ are employed by PCA for encrypting its response, the RA simply needs to include the blinded activation values in their computation by making $\widehat{X}_{t,c} = X + A_t + f_1(t{\cdot}\sigma + c) \cdot G$.

4.3 Distributing Activation Codes for Non-Revoked Devices

In ACPC, like in BCAM, activation codes can be broadcast by the CAM to all non-revoked vehicles (e.g., via satellite communication (MICHALSKI; VADEKAR., 2013)) rather than individually sent to them upon request. This eliminates the need for bidirectional connectivity between vehicles and CAMs when distributing activation codes.

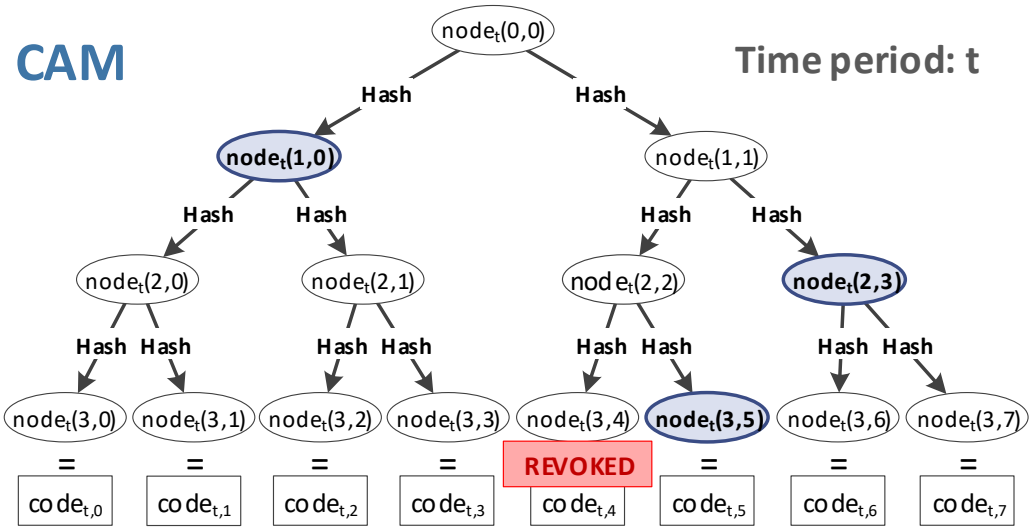
The tree's nodes that must be broadcast depend on which nodes are currently revoked/suspended, taking into account that every node of a binary hash tree can be computed from its parent. For example, given the root of the tree, all of its leaves can

be computed and, thus, all vehicles can obtain their corresponding activation codes. Hence, if no vehicle is revoked in time period t, the CAM only needs to broadcast $\texttt{node}_t(0,0)$ to allow all certificates in the system to be activated. This leads to optimal performance when distributing activation codes.

When a vehicle needs to be revoked, however, the CAM must not reveal any of the nodes in the path between the corresponding leaf and the tree's root. This prevents the computation of that leaf not only by the revoked vehicle, but also by non-revoked vehicle that might try to collude with revoked ones to allow the activation of the latter's certificates. For example, consider the tree shown in Figure 10. To revoke the vehicle whose VID is 4, the CAM would have to broadcast only the following nodes (shown in blue with ticker borders in that figure): $\texttt{node}(1,0)$, which enables the computation of leaves $\texttt{node}(3,0)$ through $\texttt{node}(3,3)$; $\texttt{node}(2,3)$, used to compute leaves $\texttt{node}(3,6)$ and $\texttt{node}(3,7)$; and the leaf $\texttt{node}(3,5)$. More generally, and as mentioned in (KUMAR; PETIT; WHYTE, 2017), when n_r vehicles out of n_t are revoked, the number of nodes included in the message broadcast by the CAM is on average $n_r \cdot \lg(n_t/n_r)$ for $1 \leqslant n_r \leqslant n_t/2$ (cf. Theorem 1 of (AIELLO; LODHA; OSTROVSKY, 1998)). Hence, albeit more expensive than the scenario in which no revocation occurs, this approach scales quite well, and is still more efficient than the individual delivery of each activation code, as in IFAL. Moreover, there are efficient methods for encoding binary hash trees such as those hereby described, so the index of each node included in the broadcast message can be represented with less than $|\text{VID}|$ bits (cf. Section 4.4 of (KUMAR; PETIT; WHYTE, 2017)), saving some bandwidth.

When compared to CRLs, using a binary hash tree does not necessarily leads to a smaller amount of data being broadcast, in particular when the number of revoked vehicles is small. Indeed, the aforementioned $O(n_r \cdot \lg(n_t/n_r))$ growth factor of hash trees for a small n_r is larger than the CRL's $O(n_r)$ growth factor discussed in Section 3.1.3. Nonetheless, hash trees scale better than CRLs for large values of n_r, since in

Figure 10: Distribution of activation tree nodes when vehicle with $VID = 4$ is revoked.



Source: the author

those cases entire branches of the tree can be omitted. Even more importantly, though, is the fact the proposed approach shifts the burden of the revocation process from vehicles to CAMs. More precisely, and as also discussed in Section 3.1.3, in SCMS the processing time required for checking whether or not a certificate is listed in the CRL is directly proportional to the number of certificates enclosed in that CRL. With activation codes, on the other hand, this additional processing cost only applies for the small time period during which pseudonym certificates remain on the CRL: after that, a revoked vehicle is simply prevented from decrypting its certificates (and the CRL remains small). Furthermore, whereas CRLs need to be stored on the vehicle's side for a long time, nodes of the activation trees do not need to be kept in storage after they are used for decrypting batches. Therefore, the additional upstream costs at CAMs are compensated by lower computational costs on vehicles. Finally, the flexibility provided by activation codes can be leveraged at any time, so the system may prefer to use CRLs while the number of revoked vehicles is small, and then shift to activation codes whenever the processing and storage costs at vehicles start becoming a burden.

4.4 Alternative Distribution of Activation Codes: Balancing Privacy and Efficiency

When activation codes are distributed via a broadcast model, as described in Section 4.3, ACPC's bandwidth usage is likely higher than what is usually obtained with SCMS-based CRLs (which are also distributable via broadcast). More precisely, when n_r vehicles out of n_t are hard-revoked, the number of nodes included in the broadcast is at most $n_r \cdot \lg(n_t/n_r)$ for $1 \leqslant n_r \leqslant n_t/2$ (AIELLO; LODHA; OSTROVSKY, 1998, Theorem 1). In comparison, CRLs as specified in the original SCMS (BRECHT et al., 2018) take roughly $2|\texttt{ls}|$ per revoked vehicle, which grows more slowly than distributing activation codes.

Even though such growth of activation trees is inherent to binary hash trees whenever hard-revocations are necessary, one important characteristic of ACPC is that vehicles do not need the whole tree for decrypting their certificates. Actually, each vehicle needs a single node of the tree, namely the one that sits on the path between its corresponding leaf and the root. When $n_r = 0$, the root would suffice and maximum efficiency is attained. When $n_r \geqslant 1$, though, vehicles could request only a branch of the activation tree that contains the required node, rather than its entirety. Interestingly, neither the request nor the response need to be authenticated: the valid activation codes enclosed in the response can be made public, and invalid node values can be detected by the vehicle simply by checking that the decrypted data does not match a valid certificate. This strikingly contrasts with CRLs, which (1) must be signed to avoid counterfeits, and (2) must be obtained in its entirety because, otherwise, the enclosing signature cannot be verified and some revoked certificates may not be identified as such.

Following such a strategy of requesting only part of the activation tree, the actual bandwidth costs for vehicles can be much smaller than what would be obtained with

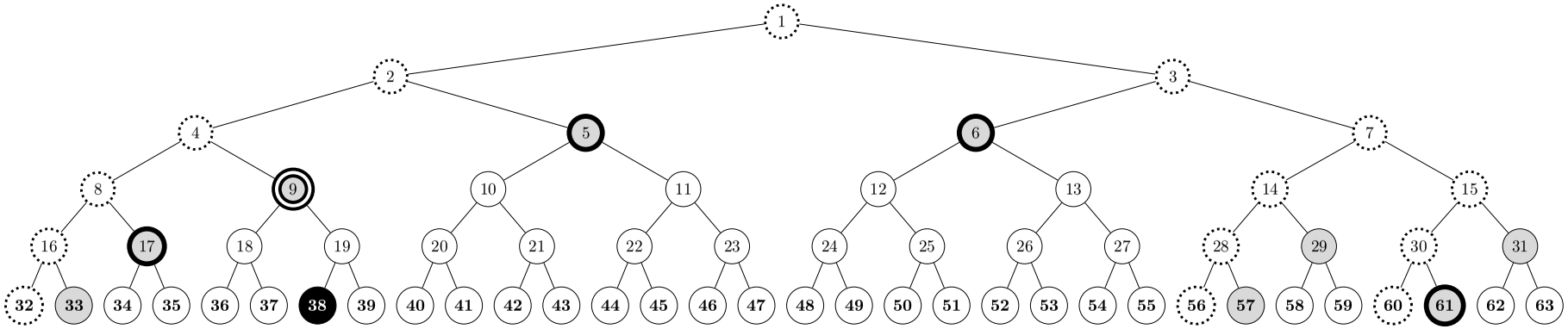
CRLs or frequent provisioning. The only potential drawback, in this case, is that the requester may end up revealing its own identity to the responder, or even to eavesdroppers, if the communication is done via an insecure channel. After all, while any vehicle might be interested in downloading the entire tree, a subset that does not include $\mathsf{node}_t(depth, count)$ would not be requested by a vehicle whose leaf is a descendant of the omitted node. Moreover, given that the choice of nodes may be personalized for each vehicle, they may rely on a unicast channel to request their activation codes.

In what follows, we present three different methods for requesting a subset of the activation tree, each one aiming at a different balance between privacy and bandwidth efficiency, namely: direct request (DR), fixed-size subset (FSS), and variable-size subset (VSS). To illustrate each approach, we use the activation trees shown in Figures 11 and 12. Both trees have depth $D = 5$ (i.e., they support $2^5 = 32$ vehicles), and depict the selection made by the vehicle associated with leaf 38 for different revocation scenarios: in Figure 11, 3 leaves are revoked (32, 56 and 60), while Figure 12 extends this scenario by additionally revoking leaves 33, 48, 49, 57 and 61, resulting in 8 revoked leaves.

4.4.1 Direct Request (DR)

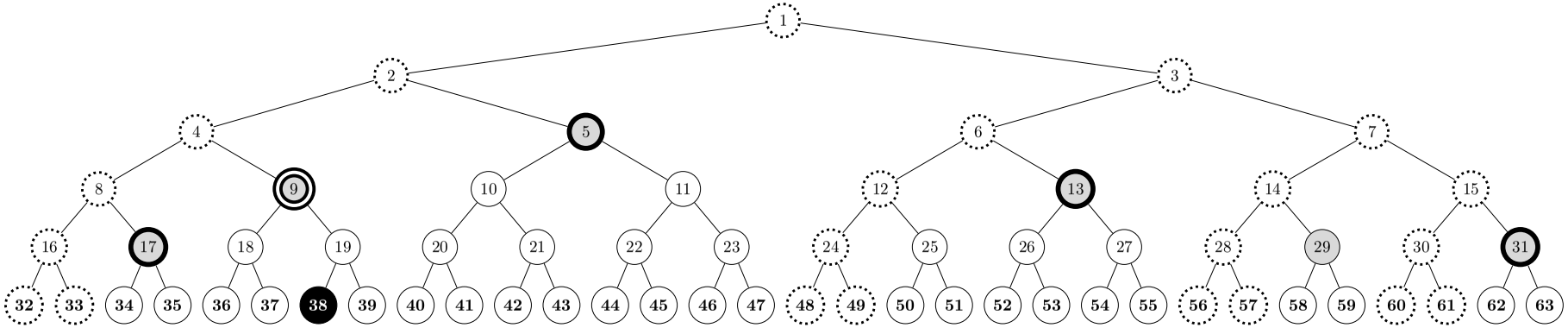
For maximum bandwidth savings, the vehicle can download a single 16-byte value, which may be the leaf that represents the vehicle or the shallowest valid node n^* on the path from this leaf to the tree root. For example, in the scenario illustrated in Figures 11 and 12, the vehicle associated with leaf 38 could request either $n^* = 9$ or node 38 directly. The former case requires knowledge of the list containing all revoked VIDs, so n^* can be determined by the vehicle; Since each VID can be represented by a D-bit string, an additional bandwidth usage of $D{\cdot}n_r$ apply if the whole list is downloaded, although some vehicles may prefer to store this list and refresh it via delta updates (COOPER, 2000). Conversely, the latter case does not involve any additional

Figure 11: Example of activation tree with depth $D = 5$, with $n_r = 3$ revoked vehicles. Vehicle 38 is requesting activation codes. Revoked vehicles are denoted with dotted border, and nodes available for picking are shaded gray. To obtain its activation code, the vehicle associated with leaf 38 (shaded black) needs to request node $n^* = 9$ (double circled). Following the FSS approach, it also includes $D - 1$ extra nodes in its request (thick border): they are randomly picked from each depth, and from lower depths (namely, depth 2) if some have zero nodes available (namely, depth 1).



Source: the author.

Figure 12: Example of activation tree with depth $D = 5$, with $n_r = 8$ revoked vehicles. Vehicle 38 is requesting activation codes. Revoked vehicles are denoted with dotted border (all associated with sibling leaves), and nodes available for picking are shaded gray. To obtain its activation code, the vehicle associated with leaf 38 (shaded black) needs to request node $n^* = 9$ (double circled). Following the FSS approach, it also includes $D - 1$ extra nodes in its request (tick borders): they are randomly picked from each depth, and from lower depths (namely, 3 and 4) if some have zero nodes available (namely, depths 1 and 5).



Source: the author.

bandwidth, since the responder may either: (1) provide n^* despite being asked for a leaf; or (2) compute the leaf on the vehicle's behalf, by performing up to D hash computations.

Albeit efficient, the privacy level provided by this approach can be as low as none. In particular, any vehicle requesting its own leaf, either willingly or due to a revoked sibling, achieves a crowd size of 1 (i.e., it would reveal its own identity). One example of the latter case refers to the vehicle associated with leaf 33 in Figure 11: since the sibling leaf 32 is revoked, there is no higher node to be requested. Other vehicles can achieve a higher privacy level, though, since the crowd size when requesting a node at depth $depth$ is given by $2^{D-depth}$. For example, if the vehicle associated with leaf 38 requests node 9 (at depth 2), the resulting crowd size is 4, since that request could be made by any vehicle interested in computing a leaf between 36 and 39.

We note that, if losing privacy is not a concern, any vehicle that desires more efficiency in receiving activation codes could make a DR. However, there are some special cases where the vehicle privacy should be forfeit. Emergency vehicles, for example, are expected to receive traffic priority, which is considered in some ITS proposals (for a literature review, see (YU et al., 2022)) but require that such vehicles could be identified. Since such vehicles can be singled out in emergency situations, they may choose to receive their activation codes by a DR, thus consuming less bandwidth in their activation process.

4.4.2 Fixed-Size Subset (FSS)

Aiming to increase the overall system privacy, each vehicle can pick several nodes in the tree as to inconspicuously blend the real requested node with additional nodes that could be picked by other vehicles. For this, each vehicle can pick as many nodes as the tree's depth D, according to the following rules (which is also represented by Algorithm 1, assuming $privacy = 0$):

I. Like in the DR approach from Section 4.4.1, pick the shallowest node n^* on the path to its leaf. Let $depth^*$ denote the depth of that node.

II. For all other pickings, try to include one random node from each other depth $depth \neq depth^*$. That will not always be possible, since some depths may not have any nodes available. In this case, randomly select extra nodes from the shallowest depths (since they provide better privacy), until D nodes are picked in this manner or there are no more nodes left to be picked.

The total data to be downloaded in this case remains quite small, comprising only $16{\cdot}D$ bytes from the list of picked nodes, together with $n_r{\cdot}D$ bytes from the list of revoked nodes' VIDs that allows the activation tree to be built so that nodes can be selected accordingly. In addition, FSS reduces the variability of the crowd size obtained by vehicles when compared with the DR method. Indeed, in any request, all vehicles obtain a crowd size larger than $2^{D-\min(depth)}$, i.e., the privacy level provided by the shallowest node available for picking. If all vehicles follow this simple strategy, they would collaboratively help to conceal vehicles whose siblings nodes were revoked: in this case, requests containing a leaf because it is the required n^* (rule I) cannot be distinguished from those where the leaf was randomly picked at depth D (rule II). This indistinguishability should prevail as long as different requests from the same vehicle remain anonymous and unlinkable to each other; otherwise, if an observer can tell that requests from the same vehicle always contain a given node n, it can infer that $n = n^*$. Fortunately, accomplishing such unlinkability is quite simple in this scenario: it is a matter of sending the request via an unauthenticated channel (e.g., HTTP), or via a secure channel that supports server-side authentication (e.g., HTTPS), while stripping the request of any information that would enable different requests to be linked to the same vehicle (e.g., see (JEONG, 2020, Sec. 5.1.2.)). If only authorized vehicles should be served, though, then the entity responsible for authorization could provide vehicles with anonymous, one-time tickets for each activation period. For example, for

3-months activation period, vehicles provisioned with 120 authorization tickets could activate all their pseudonym certificates for 30 years.

To illustrate the FSS approach, consider the following example using the tree from Figure 11. Following rule I, the vehicle associated with leaf 38 starts by including node $n^* = 9$ (from $depth^* = 3$) in its request. Then, following rule II, it randomly picks nodes from each other depth with available nodes (nodes 5, 17 and 61 from depths 2, 4 and 5, respectively). Since the tree has no available nodes at depth 1, it also selects one extra node from depth 2 (node 6), as it is the shallowest depth with available nodes. Thus, it has selected a total of $D = 5$ nodes, and the crowd size for the resulting set $\{5, 6, 9, 17, 61\}$ is then $2^3 + 2^3 + 2^2 + 2^1 + 2^0 = 23$, which corresponds to 79% of the 29 non-revoked vehicles in the system.

As another example, in the scenario shown in Figure 12, there are no nodes available at depths 1 and 5. Hence, the same vehicle associated with leaf 38 requests node $n^* = 9$ (required) from depth $depth^* = 3$, and additional nodes from depth 2 and 4 (node 5 and 17, respectively). Since it still needs two additional nodes, it also includes in its request: (1) a node from depth 3 (node 13), since it is the shallowest node available; and (2) a node randomly picked from the remaining shallowest nodes (node 31, from depth 4). In this case, the crowd size from set $\{5, 9, 13, 17, 31\}$ becomes $2^3 + 2^2 + 2^2 + 2^1 + 2^1 = 20$, or 83% of the 24 non-revoked vehicles.

It is worth noting that, despite involving a larger number of revocations, the privacy level obtained in Figure 12 is higher than the one obtained in Figure 11. This only happens, though, because where a leaf would be picked in Figure 11, a node with lower depth in the tree is picked in Figure 12, since all sibling leaves were revoked. Nevertheless, on average vehicles should expect lower privacy as the number of revocations grows. The reason is that each revocation raises the chance that nodes from lower depths become unavailable because one of its descendant leaves is among the revocation targets. In the worst case, each new revocation removes one extra node from

Algorithm 1: Pseudocode for Node Selection for FSS and VSS

```
Data: Available[0..D];  /* Array with lists of nodes available at each
      depth, from 0 to D */
Data: Path[0..D];              // Nodes in vehicle's leaf-to-root path
Data: privacy;              // Target privacy in VSS (set to 0 in FSS)
Result: Selected[0..D];     /* Array with lists of nodes picked at each
       depth, from 0 to D */
// Number of nodes to pick.  If privacy=0, toSelect=D
toSelect = NumberNodes(privacy);
if Available[0]≠null then   /* If root is available:  no revocation yet
 */
    Selected[0].AddIndex(1)                          // The root is returned
else
    for depth=1 to D do      /* Tries picking nodes from all depths */
        if Available[depth] ⊇ Path[depth] then /* Check if required node n*
         is here */
            /* A.MoveTo(B,n):  moves node n from list A to list B
               */
            Available[depth].MoveTo(Selected[depth], Path[depth]);     // Pick
             that node
            toSelect=toSelect−1; // 1 less node to pick
        else if Available[depth]≠null then
            /* A.MoveRndTo(B): moves random node from list A to
               list B                                                  */
            Available[depth].MoveRndTo(Selected[depth]) ;     // Add random
             pick
            toSelect=toSelect−1; // 1 less node to pick
        end
    end
    // If extra nodes are required,select more top-bottom
    for depth = 1 ; depth ⩽ D and toSelect > 0 ; depth = depth + 1 do
        while Available[depth]≠null and toSelect > 0 do
            Available[depth].MoveRndTo(Selected[depth]);
            toSelect = toSelect − 1;
        end
    end
end
```

the activation tree's lowest depth (i.e., those that provide a larger crowd size). As a result, all nodes from depths lower than $depth = \lceil \lg(n_r) \rceil$ become unavailable, forcing the vehicle to pick $depth + 1$ nodes from depth $depth + 1$. The crowd size obtained is, thus, at least $(\lceil \lg(n_r) \rceil + 1) \cdot 2^{D-1}/n_r$, which corresponds to the crowd size provided by such lower depth nodes (i.e., ignoring the privacy added by pickings from higher depths).

4.4.3 Variable-Size Subset (VSS)

Finally, as an extension of the FSS approach, vehicles could increase their own privacy by randomly selecting extra nodes from the activation tree's shallowest depths, including those nodes in their requests. This case is covered by Algorithm 1 with $privacy > 0$, indicating that extra privacy is desired. For example, suppose that the vehicle associated with leaf 38 wants to increase the crowd size in the scenario from Figure 11. Namely, assume its goal is to be confused with at least 83% of non-revoked nodes, as it was the case with the FSS method in the scenario depicted in Figure 12. For this, that vehicle can select $6 > D$ nodes, increasing the crowd size by 2 vehicles by including either node 29 or node 31 in its own request. The resulting crowd size is then 25, or 86% of all non-revoked nodes.

In general, it is easy to compute the number of additional nodes to be picked beyond D if the list of nodes available at each depth is available. Namely, as discussed in Section 4.4.2, each extra node from depth $depth$ adds $2^{D-depth}$ to the crowd size. Similarly to FSS, though, the actual privacy requires that every request is unlinkable to each other, or different requests might allow the responder infer which vehicle is making them.

The bandwidth cost when using this strategy depends on the desired crowd size, as well as on the number of nodes available at the shallowest depths of the tree. However, as discussed in Section 6.2, our simulations show that this approach scales quite well

for realistic sizes of activation trees and number of revocations.

4.5 Summary

In this chapter, we presented the proposed solution to improve certificate revocation, called Activation Codes for Pseudonym Certificates (ACPC). The solution describes the usage of activation codes to deny the usage of previously acquired pseudonym certificates if the codes are not sent by the system. It allows the removal of certificates from the Certificate Revocation List (CRL) after their expiration since the vehicles will not be able to use non-activate certificates. We showed how to generate activation codes and issue certificate, interleaving them with the butterfly key expansion. We then presented how to broadcast the activation codes to non-revoked vehicles and also argued how to better distribute activation codes in the unicast model. For that, we detailed three different methods (namely, direct request – DR –, fixed-size subset – FSS–, and variable-size subset –VSS), each achieving a different balance between efficiency and privacy.

5 SECURITY ANALYSIS

In this chapter, we evaluate the security of ACPC's certificate issuance and revocation procedures. The discussion follows an informal approach, building upon the security properties of SCMS itself and on standard assumptions, in particular the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP). Even though providing formal security proofs (e.g., based on the Game-Playing Technique (BELLARE; ROGAWAY, 2006)) would be even better, doing so would require building formal proofs for SCMS itself, since the latter only displays an informal (albeit, according to our analysis, quite sound) security discussion. Therefore, we leave the task of building a formal security model and proving the security of ACPC (and, consequently, of SCMS) in this model as a topic for future work.

5.1 Security of the Certificate Issuance Process

In ACPC, a collusion between CAM and PCA (resp. RA) reveals as much information as the PCA (resp. RA) had available in the original SCMS. Indeed, if we remove the influence of A_t over the public cocoon keys computed as described in Section 4.2, the result matches the public cocoon keys in SCMS. Therefore, a collusion with the CAM can only remove the entropy introduced by this entity, while still preserving the security and privacy properties discussed in Section 3.1.2 for SCMS itself.

In particular, a vehicle's private caterpillar key x (either s for signing or e for encrypting) remains protected by the ECDLP during the whole execution of the protocol.

Hence, RA, PCA and CAM are unable to recover the signature or decryption private keys derived from it, even if they collude. Unlinkability among certificates is similarly preserved, as long as the RA and PCA do not collude: the shuffling done by the RA still hides from the PCA any relationship between certificate requests intended for a same vehicle; meanwhile, the PCA's encrypted response prevents anyone but the appropriate vehicle from learning $cert_i$. Interestingly, even if the CAM delivers the blinded activation values A_t via an unencrypted channel, an eavesdropper (e.g., the PCA) would remain unable to identify the cocoon encryption keys to which they are associated: after all, if two keys $\hat{E}_1$ and $\hat{E}_2$ belonging to the same vehicle are indistinguishable from random points (and, hence, unlinkable), so are the $\hat{E}_1 + A$ and $\hat{E}_2 + A$ shifted keys resulting from the ACPC certificate provisioning process. Finally, since the PCA's signature performed during the butterfly key expansion process grants vehicles the ability to verify whether or not the received certificates were generated in a fair manner, MitM attacks (e.g., by the RA) are averted.

It is worth mentioning that the aforementioned CAM's inability to create a new threat via collusion with other entities is not just a fortunate coincidence. Actually, the activation codes $\mathsf{code}_{t,\mathrm{VID}}$ are the only information initially kept secret by the CAM and, thus, that could be contributed in such collusion. Since those codes are periodically broadcast aiming to allow vehicles to activate their certificates, though, by design such a disclosure of the activation tree should not bring negative impacts on the system's security or privacy. Consequently, a "private disclosure" during a collusion is expected to have an equivalent result.

5.2 Security of the Revocation Procedure

The security of ACPC's revocation procedure relies on the first preimage resistance of the hash function employed for the construction of activation trees, as well as the proper disclosure of its nodes by the CAM. In principle, following a dishonest-

if-allowed model, this means that the choice of a secure hash function is enough to enforce revocation. At least this should be the case if we assume that the system's authorities would gain nothing by illegitimately un-revoking vehicles, i.e., without authorization from the MA. Nonetheless, it is useful to evaluate what happens when one of the system's entities is compromised and, as a result, its capabilities are misused aiming to allow the activation of revoked devices.

On the one hand, a rogue/compromised CAM could disclose the roots of every activation tree to all vehicles, even revoked ones, allowing all certificates in the system to be activated. This would not give the attacker any advantage, though, besides disrupting the system's ability to revoke devices in an efficient manner. In particular, in consonance with the discussion in Section 5.1, this would not grant the CAM or any other entity the ability to track devices. Consequently, in a dishonest-if-allowed scenario, it is unlikely that the CAM itself would go rogue and engage in such malicious activity. Furthermore, if the CAM's storage is somehow compromised, the leakage of codes can still be contained by keeping the revoked vehicles' data in CRLs, just like in the original SCMS; alternatively, the CAM itself can be revoked so no certificate under its responsibility is considered valid, thus avoiding the growth of CRLs' size. Hence, the attack would not result in any catastrophic security breach, but only nullify the performance gains provided by activation codes.

On the other hand, a security breach at the RA or PCA should not reveal any information about activation codes. The reason is that these entities never learn those codes, which are only known by the CAM. Nevertheless, if any of these entities goes rogue or is compromised at a level that allows its behavior to be controlled by attackers, it can provide valid certificates to revoked vehicles independently of activation codes. Specifically, a dishonest PCA can always issue new pseudonym certificates for vehicles, including revoked ones, at least until the PCA itself is revoked. A compromised RA could act similarly, e.g., by requesting pseudonym certificates for a non-revoked

VID_d, and then delivering those certificates to a revoked vehicle whose identifier is $\text{VID}_r \neq \text{VID}_d$. Such misbehavior is likely to go unnoticed because the certificates do not carry any VID on them, as it would break the unlinkability provided by the pseudonyms. In addition, if VID_d corresponds to a valid vehicle whose pseudonym certificates have not been requested yet, the CAM would not be able to notice the fraud by the RA: from the CAM's perspective, this is simply a new vehicle in the system.

Actually, even if VID_d has already been requested in the past, it would be difficult for the CAM to use this knowledge to prevent such attack. The reason is that, if the CAM is configured to refuse a second request for the same VID_d, management issues are likely to arise. For example, the processing of the initial request for VID_d may naturally fail, so an honest RA would actually need to send a second request referring to the same VID_d. As another example, a dishonest RA might abuse this process by performing a "denial-of-certification" attack: the RA queries the CAM requesting the caterpillar keys for a non-revoked VID_d, but it does not execute the pseudonym certificate issuance process; future requests referring to VID_d, potentially by honest RAs, would then fail.

These observations indicate that, even if a rogue RA or PCA never gains access to activation codes, their roles in the system still enable them to provide valid certificates for revoked vehicles. Actually, a similar discussion also applies to the original BCAM protocol, in which a rogue RA or PCA could provision revoked vehicles with the PCA-encrypted certificates, before they are once again encrypted by the CAM. Even though ACPC's approach of ensuring that only the CAM is able to distribute activation codes does not actually prevent such threats, it was adopted because it does reduce the system's attack surface. For example, if RA and/or PCA store the (PCA-encrypted) certificates, a data breach disclosing such certificates for a vehicle that is now revoked does not create any security concern, since that vehicle remains unable to decrypt them without the corresponding activation codes.

5.3 Disaster Recovery

In ACPC, the CAM is assumed to follow a "dishonest-if-allowed" model, similarly to the RA and PCA in SCMS (and to the CAM in BCAM). Therefore, it is not expected to simply broadcast wrong codes or refuse to broadcast activation codes for non-revoked vehicles, in particular because this would only allow the CAM to disrupt the system, without giving it any actual benefit (e.g., it would remain unable to track vehicles). Furthermore, such misbehavior could be easily detected: the absence of broadcast messages can be noticed by any system entity, while wrong codes can be detected by the RA by verifying whether or not a blinded activation value originally received from the CAM was generated from the disclosed activation code. Such verification by the RA can be performed periodically, via random sampling, or could be triggered by a complaint from a non-revoked vehicle unable to decrypt its own certificates.

An honest CAM may, nevertheless, become target of denial of service (DoS) attacks aiming to prevent the delivery of activation codes, thus impairing the availability of the whole system. Fortunately, however, there are many possible approaches to mitigate such threat. For example, even though the generation of the activation tree is expected to be centralized at each CAM, the distribution of activation codes can actually be shared with multiple entities, including RAs, PCAs, vehicles and roadside units. After all, the trees disclosed by CAMs do not carry any secret information (otherwise, they would not be broadcast) and can be signed by the responsible CAM to ensure its authenticity. In addition, catastrophic events such the CAM losing access to all activation codes can be addressed by means of standard disaster recovery approaches, such as saving the roots of the activation trees in different places and protecting them via secret sharing mechanisms (SHAMIR, 1979) (AGRAWAL et al., 2009).

5.3.1 Support for Soft Revocation with Vehicle-Side HSMs

In BCAM, vehicles are assumed to be equipped with a hardware security module (HSM). This enables what is called a "soft-revocation" mechanism: instead of asking the CAM to omit nodes from the binary tree, the Misbehavior Authority could periodically issue a soft-revocation list (SRL) containing identifiers of revoked vehicles; as a result, the HSM of vehicles listed in the SRL simply refuse to compute the decryption keys for the corresponding certificates. A similar feature can be provided in ACPC if, like in BCAM, the HSM exports a CAM-encrypted symmetric key vk, which is included in the vehicle's request for pseudonym certificates. Then, by computing the blinded activation values as $f_a(vk, \mathtt{code}_{t,\mathrm{VID}}, t, \mathrm{VID})\cdot G$, the CAM ensures that the HSM is the only entity capable of decrypting certificates, since it is the only component with access to vk.

The advantage of this approach is that it potentially leads to smaller messages broadcast by CAMs. After all, as discussed in Section 4.3, the (hard) revocation of vehicles forces the CAM to disclose multiple nodes of the revocation tree, rather than only its root. If, however, those vehicles are known to be soft-revoked due to a compliant HSM, the activation tree's root can be disclosed without negative impacts to the system's security.

Albeit interesting, such a soft-revocation mechanism is considered optional in ACPC because some vehicles may actually not have an HSM installed, or the HSM might be compromised by a crafty attacker, rendering the SRL useless. In those cases, the CAM would be obliged to fall back to the conventional, more reliable process of omitting nodes from the binary hash tree, so some CAM may actually prefer to default to such a safer approach.

5.4 Summary

In this chapter, we analyzed the security of our proposed solution, showing that it does not introduce new vulnerabilities to SCMS. The evaluation is made by comparing which values are made public in the certificate issuance and revocation, when considering the collusion of the CAM with any of the certification authorities: the RA or the PCA. We argued that, if the CAM colludes, the vehicle privacy is not lost, and the other authority only receives the capability of activating the certificates as if the CAM ceased to exist. We also consider the scenario in which the CAM is corrupted after the generation of the activation codes in a way that its usage can be reverted to the original SCMS scheme without loss of security. Furthermore, in case of vehicles equipped with a hardware security module, soft revocation is possible, improving the distribution of the activation codes.

6 COMPARISON WITH RELATED WORKS

The features proposed in ACPC modify the relationships among the system authorities with the vehicles by holding access to the pseudonym certificates. These changes not only improve the management of the Certificate Revocation List (CRL), but also improves the privacy provided to the vehicles during the issuance. Given the diversity of properties each VPKI support, in this chapter we discuss the differences in security and efficiency between ACPC and the VPKIs described in Chapter 3 (namely, SCMS, C-ITS, IFAL and BCAM).

Since ACPC was built upon the SCMS, most of security aspects are maintained regarding the system authorities present in both architectures. With the inclusion of the CAM, however, we have an additional network interface that requires bandwidth when issuing certificates. But, as argued in Chapter 5, a collusion between the CAM with the PCA or the RA would reveal as much information as SCMS itself. Although also built upon SCMS, BCAM does not share this same feature, since the CAM in the latter would be able to link pseudonym certificates to a vehicle.

When compared to IFAL and C-ITS, ACPC differs mainly in the threat model of the system authorities. IFAL and C-ITS allow an “honest-but-curious” PCA to link several certificates to a same device. Since they create certificates without linkage information, the capability to link a pseudonym to the vehicle’s identity was given to a single authority (namely, the PCA). However, as ACPC is based on SCMS, linkage information can only be obtained if the PCA and the RA collude, which is expected

only when a vehicle should be revoked.

One advantage of IFAL over C-ITS is the issuance of pseudonym certificates beforehand. By moving most of the transmission costs out of band, the bandwidth costs are greatly reduced in real-time, as IFAL's activation codes are much smaller than actual certificates. However, ACPC allows vehicles to obtain activation codes much more efficiently than IFAL's strategy. By using binary hash trees to create activation codes, the CAM can broadcast them directly to the vehicles or to caching units, instead of relying that vehicles individually request them.

Architecturally, ACPC shares more similarities with BCAM than with IFAL, in particular because both BCAM and our proposal use binary hash trees for the distribution of activation codes. However, ACPC provides better security, for at least two reasons. First and foremost, the fact that the CAM does not receive certificates from the RA prevents the former from learning which PCA-encrypted certificates belong to a same device; therefore, and unlike BCAM, a collusion between CAM and PCA does not allow those entities to track vehicles. Second, as discussed at the end of Section 5.2, ACPC also reduces the revocation procedure's attack surface by a rogue RA or PCA, so it is able to protect the vehicle's privacy even if PCA and CAM collude or are compromised.

In the following sections, we compare ACPC with the other VPKI schemes in terms of processing costs and bandwidth usage, considering both the broadcast and the unicast models of distribution of activation codes. Additionally, aiming to analyze the trade-off in privacy for each of the alternative distributions strategies from Section 4.4, we developed a simulator for activation trees that is able to revoke a configurable number of leaves from an activation tree. Such revocations can be done either randomly (for simulating an average case), or by targeting leaves that are descendants of the lowest-depth nodes, which then become unavailable for picking (which corresponds to the worst case). The simulator then calculates the size of the whole activation tree, as

well as the bandwidth usage and crowd size obtained with each node selection strategy. The results from this analysis are discussed in the following section. For better reproducibility of our experiments, the simulator's source code (in Java) is available for download at Simplicio, Silva and Cominetti (2019).

6.1 Processing Costs

To evaluate the processing overheads incurred by ACPC, it is important to consider every component of the system. In what follows, we start by evaluating the processing costs at the CAM and vehicles, for which the benefits of ACPC are more noticeable, and then we discuss how the PCA and RA are affected.

One improvement of ACPC over BCAM is that, in the latter, the CAM is responsible for encrypting the batches received from the RA, after they are delivered by the PCA. Consequently, the CAM ends up encrypting certificates that have already been encrypted by the PCA: the latter process is intended to preserve the vehicles' privacy, and the former is executed for the purpose of preventing their certificates from being activated in case of revocation. The proposed approach integrates both purposes into a single encryption process, performed by the PCA, which was already present in SCMS itself.

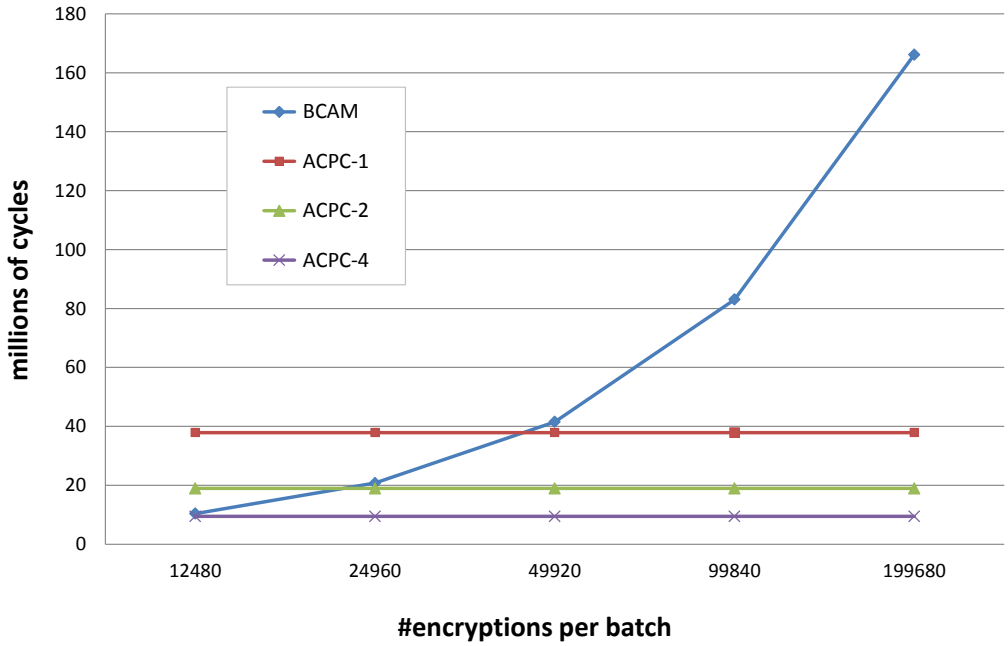
More precisely, in BCAM the number of encryptions performed by the CAM grows proportionally to the size of the batches, which depends on the size of *pkg* and on the number of certificates per time period (σ). In comparison, ACPC's design is such that each activation period requires one PRF execution combined with an elliptic curve scalar multiplication, for computing the corresponding blinded activation values $A_t = f_a(\mathsf{code}_{t,\mathrm{VID}}, t, \mathrm{VID}) \cdot G$. Therefore, many symmetric encryption operations can be replaced by a single computation of a blinded activation value. Since scalar multiplications over elliptic curves are known to be considerably more expensive than

symmetric encryption, however, the actual benefits of ACPC depend on the system parameters: for a given value of $|pkg|$, the larger the number of time periods covered by each activation period (α), the larger the number of encryptions replaced by the computation of a blinded activation value and, hence, the better the processing times of ACPC when compared to BCAM.

To give some concrete numbers, we have implemented our proposal using the RELIC cryptography library version 0.4.1 (ARANHA; GOUVÊA, 2018) running on an Intel i5 4570 processor. In this setting, the computation of one blinded activation value takes 242,822 cycles. This is approximately 300 times the cost of one single AES encryption, which takes only 832 cycles. Based on these numbers, Figure 13 shows the total processing costs at the CAM when we assume pseudonym certificate batches that span a period of 3-years, which corresponds to 156 one-week long time periods. The smallest number of encryptions for BCAM, 12,480, refers to a scenario where $\sigma = 20$ and $|pkg| = 512$ bits, meaning that $156 \cdot 20 = 3120$ pseudonym certificates are encrypted by the CAM and each encryption takes $512/128 = 4$ calls to a 128-bit block cipher such as AES. The largest number of encryptions, 199,680, is obtained when $\sigma = 40$ and $|pkg| = 4096$ bits, so the encryption of each of the $156 * 20 = 6240$ certificates requires $4096/128 = 32$ calls to the underlying 128-bit block cipher. As observed in this figure, the performance of ACPC is comparable to BCAM's, and usually surpasses it, in particular for larger numbers of α. For instance, when each activation period spans 4 certificate time periods, only $156/4 = 39$ blinded activation value are computed, so the corresponding ACPC-4 curve always remains below BCAM's curve.

Even more importantly, the fact that the CAM does not re-encrypt batches in ACPC means that the vehicles do not need to remove this additional encryption layer when processing the received pseudonym certificates. Instead, the only decryption operation performed at the vehicle's side is the one originally present in SCMS itself, which reverses the encryption done by the PCA. Therefore, the total overhead

Figure 13: Processing costs at the Certificate Access Manager (CAM) for BCAM and ACPC. We denote by ACPC-α the setting in which the number of time periods covered by each activation period is α.



Source: the author.

incurred by ACPC refers to the computation of one (unblinded) activation value $f_a(\mathsf{code}_{t,\mathrm{VID}}, t, \mathrm{VID})$ per activation period, so it can be included in the computation of the decryption key $\hat{e}_{t,c}$. When compared to BCAM, this means that $|pkg|/128$ encryptions are replaced by τ/α invocations of a PRF at the vehicle's side. Even if we consider the scenario from Figure 13 in which the benefits of ACPC are less pronounced at the CAM's side (namely, $\alpha = 1$ and $|pkg| = 512$), this still means that ACPC trades 12,480 decryptions for 156 PRF executions. In practice, assuming that one PRF execution takes roughly as much processing as one decryption, this means that the overhead added by ACPC's activation codes is approximately 1/80 of the overhead incurred by BCAM at the vehicles' side.

Finally, the usage of activation codes is transparent from the PCA's perspective, so its processing costs remain unchanged. For the RA, in turn, a single elliptic curve addition per activation period is necessary, to compute $E + A_t$, from which the corre-

sponding public cocoon keys $\hat{E}_{t,c}$ can be obtained. Therefore, the overhead incurred by ACPC at the RA is expected to be negligible.

6.2 Bandwidth Usage

While we expect the use of activation codes to reduce the required bandwidth to issue certificates, the inclusion of a new authority (namely, the CAM) also creates a new network interface to the system. In this section, we detail the costs of data exchanged in both the system authorities and during the broadcast of activation codes. We also compare the alternative distribution methods for unicast communication considering the data required to distribute CRLs in the original SCMS scheme.

6.2.1 System Authorities

Whether pseudonym certificates are issued with BCAM or with ACPC, the amount of data exchanged in the vehicle-RA and RA-PCA interfaces remains basically the same as in the original SCMS (see Figure 9, which shows all interfaces involved in the issuance process). In the RA-CAM interface, on the other hand, ACPC ends up saving bandwidth: whereas BCAM exchanges (large) batches of pseudonym certificates via this interface, in ACPC the CAM simply sends elliptic curve points to the RA upon request. More concretely, suppose that SCMS adopts implicit certificates, which are usually shorter that explicit ones, and employs the optimizations described in (SIMPLICIO et al., 2018c), which removes the need of signing the encrypted package pkg. In that case, each PCA-encrypted response $pkg = Enc\left(\hat{E}_i; \{cert_i, sig_i\}\right)$ is expected to be such that $|pkg| \geqslant 6k$ bits for a security level of k: (1) $2k$ bits for the implicit credential V_i enclosed in $cert_i$, since it corresponds to an elliptic curve point; (2) $2k$ bits for the random elliptic curve point that is generated during any ECIES-based encryption; (3) $2k$ bits for the signature sig_i; and (4) some extra bits for the certificate's metadata (e.g., a linkage value and an expiration date), as well as for the authentication tag resulting

from the ECIES-based encryption, which we ignore in our calculations. Hence, for a batch size $\beta = \tau \cdot \sigma$, in BCAM the total bandwidth usage at the RA and CAM would be $2 \cdot (6k) \cdot (\tau \cdot \sigma)$ bits, where the "2" multiplication factor is due to the fact that this cost is paid both downstream and upstream: the RA sends PCA-encrypted batches to the CAM, and then receives CAM-encrypted batches.

In comparison, in ACPC the CAM simply sends τ/α blinded activation values to the RA, where $\alpha \geqslant 1$ is the number of time periods covered by each activation period. Since each blinded activation value A is an elliptic curve point (i.e., $2k$ bits), this corresponds to an upstream (resp. downstream) cost at the CAM (resp. RA) of $2k \cdot \tau/\alpha$ bits. Therefore, ACPC is expected to use approximately $1/(6\sigma \cdot \alpha)$ of the bandwidth required by BCAM during the pseudonym certificate issuance process. To give a numeric example, suppose that the system is configured in such a manner that 20 certificates are valid per time period (as in (KUMAR; PETIT; WHYTE, 2017)), and each activation period covers 4 time periods (e.g., for certificates valid for a week, the activation period would correspond to a month). In this case, issuing batches of certificates with ACPC would take 1/480 of the bandwidth required by BCAM at the RA-CAM interface.

6.2.2 Distribution of Activation Codes

Regarding the distribution of activation codes by broadcast, ACPC provides bandwidth savings when compared to BCAM despite also using a binary hash tree as underlying data structure for activation codes. This happens because, by integrating security strings into the activation trees, the nodes of those trees can be 128-bit long while still preserving a 128-bit security level, despite the number of revoked devices. When compared to BCAM, which uses 256-bit nodes for achieving the same security level (see Appendix A), this represents a 50% bandwidth gain for the distribution of activation trees.

To compare the costs of the DR, FSS and VSS approaches to distribute activation codes using unicast communication, we first analyze the amount of data exchanged in each method. For that, we first assume that the encoding of ACPC's activation tree corresponds to the list of D-bit revoked IDs (i.e., by removing the revoked leaves from the complete tree), which translates to $D{\cdot}n_r$ bits. Similarly, any requested node is represented using a D-bit string, whereas the value of such nodes consists of 128-bit strings (assuming ACPC is running with a 128-bit security level).

In the DR approach, vehicles simply send a D-bit request and receive a 128-bit response. In the worst-case scenario, where vehicles wish to request an internal node but do not have the activation tree's encoding locally cached, an additional download overhead of $D{\cdot}n_r$ bits would apply.

Conversely, in the FSS scheme, vehicles are required to obtain the activation tree's encoding, once again at a cost of downloading at most $D{\cdot}n_r$ bits (assuming it is not already cached). Then, the vehicle requests and receives D nodes, which translates to download and upload costs of $128 \cdot D$ and D^2 bits, respectively.

Lastly, in the VSS approach, the number of requested nodes depends on the desired privacy settings and on the distribution of revocations among the tree's leaves, since, if available, picking shallower nodes results in a larger crowd size for achieving the expected privacy level. Figure 14 shows the number of nodes required by a vehicle for $D = 40$ and a target crowd size of $0.1 \cdot n_t$; this means that the system supports roughly 1 trillion vehicles, and each vehicle can be confused with 10% of its peers when requesting its own activation codes. For each number of revocations $0 < n_r \leqslant 50,000$, we consider the worst-possible distribution of revoked nodes, as well as the average case for 10,000 random samples. As observed in this figure, the average number of requested nodes grows approximately logarithmically with n_r, which means at a pace smaller than the $n_r \cdot \lg(n_t/n_r)$ upper limit observed for the whole activation tree. The reason for this behavior is that nodes from shallower depths, which provide a higher

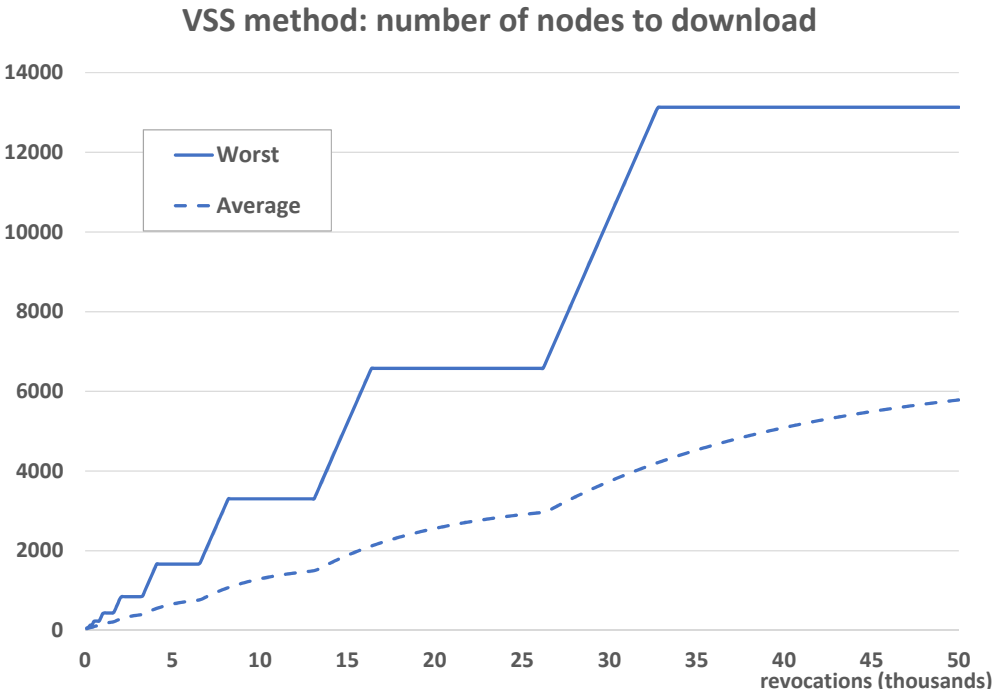
crowd size, are removed from the activation tree at a logarithmic pace; for example, to remove all nodes from depth $depth$, at least 2^{depth} leaves must be revoked. Whenever a depth $depth$ is depleted of nodes, it takes two nodes from depth $(depth + 1)$ to get the same crowd size that would be obtained with a node from depth $depth$.

In the specific case of our simulations, the request needs to include between 0.5–1% of activation tree's nodes. Figure 15 shows the corresponding bandwidth costs for VSS, considering not only the requested nodes but also the download of the entire activation tree's encoding (assuming it is not cached). For easy reference of the savings obtained by caching the tree's encoding, we also depict the downstream costs for this piece of data alone. All in all, and although the download costs in this case are 2 to 3 times larger than in the FSS approach, the total amount of data remains small: even for 50,000 revocations, VSS's extra privacy incurs a download size below 450 KiB in the worst case, and below 335 KiB on average, of which 244 KiB refers to the activation tree's encoding.

To give an overall picture of the bandwidth costs of each distribution approach, we now compare them with all the revocation methods hereby discussed, namely ACPC (including DR, FSS, VSS, and the broadcast methods), C-ITS, and the original SCMS with CRLs. Since IFAL and BCAM present the same growth as DR and the broadcast method, respectively, they are not included in this comparison. In Figures 16 and 17, we compare the downstream and upstream costs per vehicle for different values of n_r, in each activation period. The corresponding costs at the server-side can then be obtained by multiplying those numbers by the system's fleet size. Notice, though, that such costs on the PKI infrastructure would correspond to an upper limit in ACPC, where, unlike C-ITS, the burden of distribution can be shared with caching units.

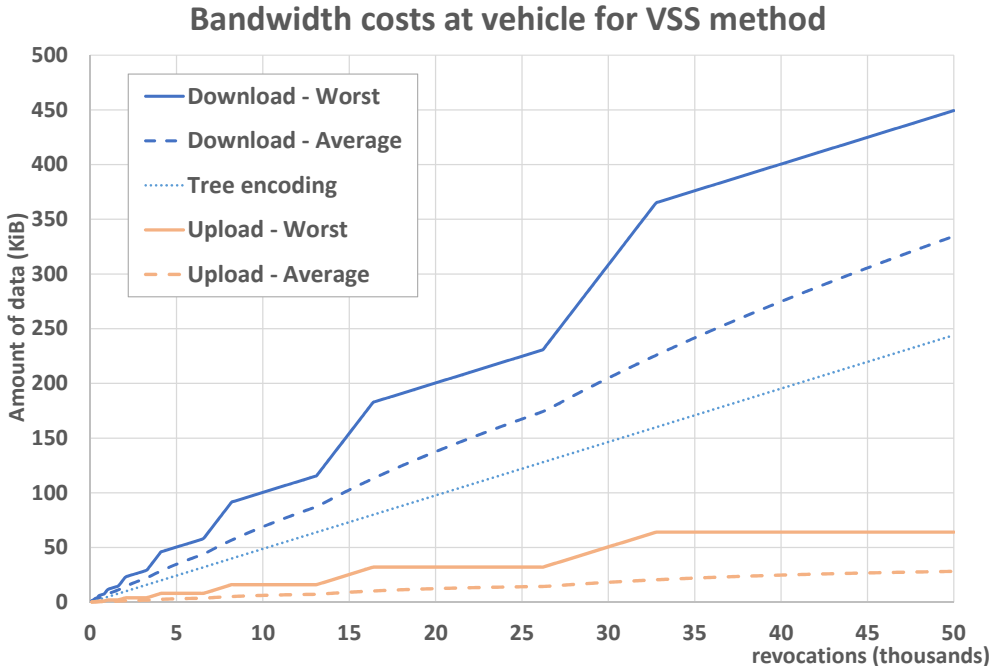
For ACPC's broadcast approach, we used our simulator to obtain the average number of nodes in the entire activation tree; the corresponding download costs are then computed simply as 128 bits per node, plus 40 bits per revoked ID (from which the IDs

Figure 14: Number of nodes required to achieve 10% privacy in the VSS method, for 0 to 50,000 revocations.



Source: the author.

Figure 15: Bandwidth (download and upload) costs at vehicle to achieve 10% privacy in the VSS method, for 0 to 50,000 revocations.



Source: the author.

of the nodes can be inferred). For the C-ITS certificates, we consider two cases when computing the underlying downstream costs: (1) ignoring any overhead, so the only cost refers to the 117-bytes certificate payload; and (2) estimating the actual C-ITS certificate response message based on its specification (ETSI, 2021, Sec. 6.2.3.3.2), which means 265 bytes for each certificate (see Appendix B for details on how this number was estimated). Similarly, for the upstream cost analysis, we estimate the size of a C-ITS certificate request message to be around 505 bytes (see Appendix B).

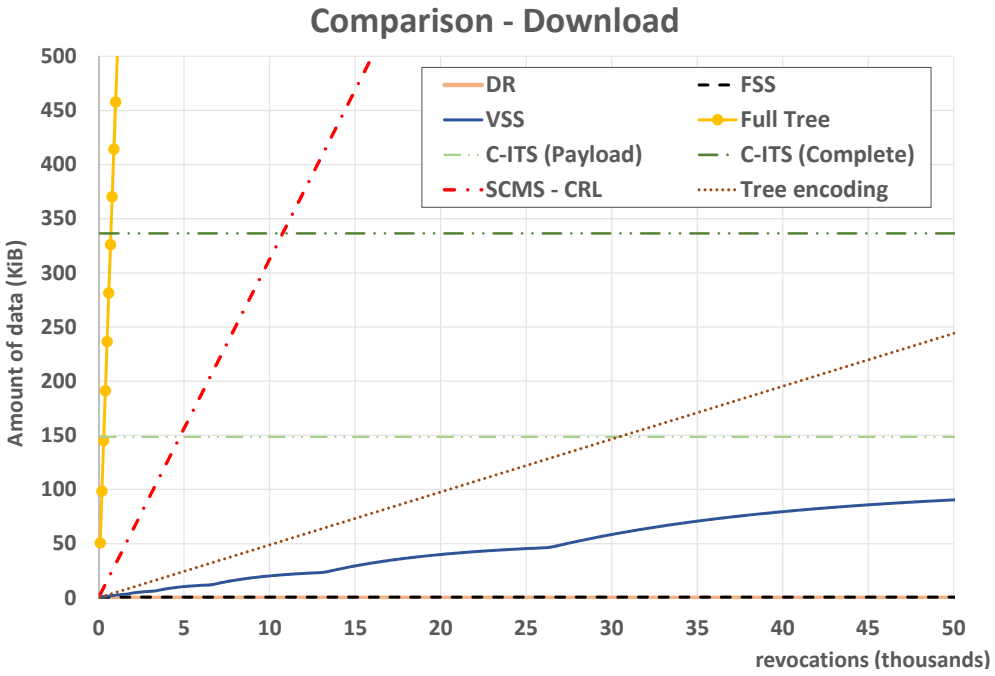
Concerning the downstream costs per vehicle, Figure 16 shows that downloading the entire activation tree results in low scalability. Specifically, this distribution method becomes the most expensive method after approximately 400 revocations, which reinforces the need of efficient broadcast mechanisms and caching when such method is adopted. SCMS with CRLs is the second least scalable approach. Even though such a burden may be tackled via delta-CRLs, so only updates need to be downloaded, this would not address the other drawbacks of this approach, such as the lack of flexibility and the overhead processing when verifying certificates. In addition, delta-updates could be similarly employed to the download of the activation tree encoding in the DR, FSS and VSS methods, considerably reducing their downstream costs. Indeed, Figure 16 shows that the amount of data downloaded for the DR, FSS and VSS approaches remain much lower than any alternative strategy when the tree's encoding is not taken into account. In addition, even if the tree encoding is downloaded in every activation period, DR, FSS and VSS remain quite competitive: all three methods are more efficient than C-ITS in a scenario with less than 50 thousand revocations; if compared with C-ITS's payload only, then the break-even for DR, FSS and VSS would be around 22000, 22000 and 30000 revocations, respectively.

In terms of upload costs on the vehicle-side, a first remark is that obtaining the full ACPC activation tree or CRLs precludes any specific need for upstream communication. After all, both data structures can be obtained through a generic request or

via broadcast, with negligible upstream cost. For this reason, Figure 17 only depicts the costs for C-ITS, DR, FSS and VSS. As observed in this figure, C-ITS is quite inefficient regarding this metric. This is explained by the combination of two facts: (1) for better privacy, C-ITS requires vehicles to generate one request per pseudonym certificate, meaning up to 1,300 requests per period; and (2) such requests are not too small, since each one contains a public key encrypted and signed by the vehicle, and the entire request is also encrypted and signed. In comparison, the requests in the DR methods take a single 40-bits node identifier (i.e., 5 bytes), while FSS involves $D = 40$ of such identifiers (i.e., 200 bytes). Even though the upstream cost in VSS is not as negligible, for the simulated privacy level of 10% it still remains close to 64 KiB, even if we consider the worst-case scenario for 50,000 revocations (the maximum number shown in this figure). Since this corresponds to 10% of the C-ITS approach, we can conclude that ACPC is considerably more appealing than C-ITS when the goal is avoiding upstream costs.

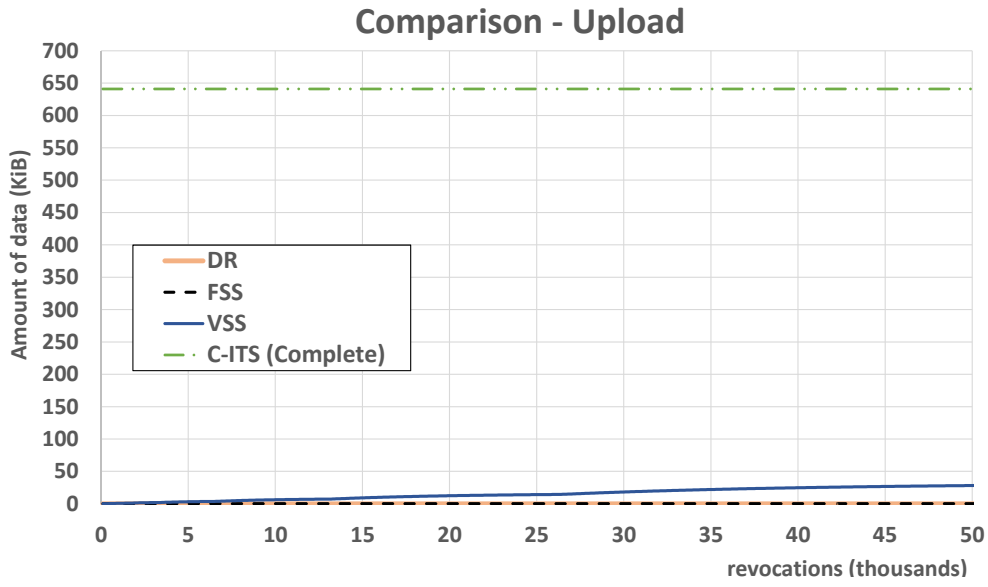
Finally, we note that the vehicle-side upstream costs of C-ITS could be reduced if it is combined with butterfly keys (BRECHT et al., 2018; SIMPLICIO et al., 2018c), as done in ACPC itself. Basically, this enables vehicles to offload part of those costs to a proxy, called a Registration Authority (RA) in the SCMS architecture (BRECHT et al., 2018). As a result, an arbitrarily large batch of pseudonym certificates can be generated from a single request containing a 96-byte payload (or 48 bytes if the unified butterfly optimization is adopted (SIMPLICIO et al., 2018c)). Nevertheless, the resulting VPKI's bandwidth costs are not reduced, since the proxy expands the vehicle's request into as many individual requests as required in C-ITS without butterfly keys. In addition, to preserve privacy, the (unified) butterfly key mechanism requires a location obscurer proxy to decouple the vehicle's identity and location. Under these settings, adopting ACPC with the DR approach would provide as much privacy as C-ITS, but would involve a 5-byte request sent via an unauthenticated channel rather than a 48- or

Figure 16: Comparing vehicle-side download costs in each activation period, for different V2X-oriented revocation methods: SCMS with ACPC (DR, FSS, VSS, and broadcast), SCMS with CRLs, and C-ITS. VSS is configured to provide 10% privacy.



Source: the author.

Figure 17: Comparing vehicle-side upload costs in each activation period, for different V2X-oriented revocation methods: SCMS with ACPC (DR, FSS, VSS), and C-ITS. VSS is configured to provide 10% privacy. SCMS with CRLs and ACPC with broadcast are not presented as they do not require any upstream communication.



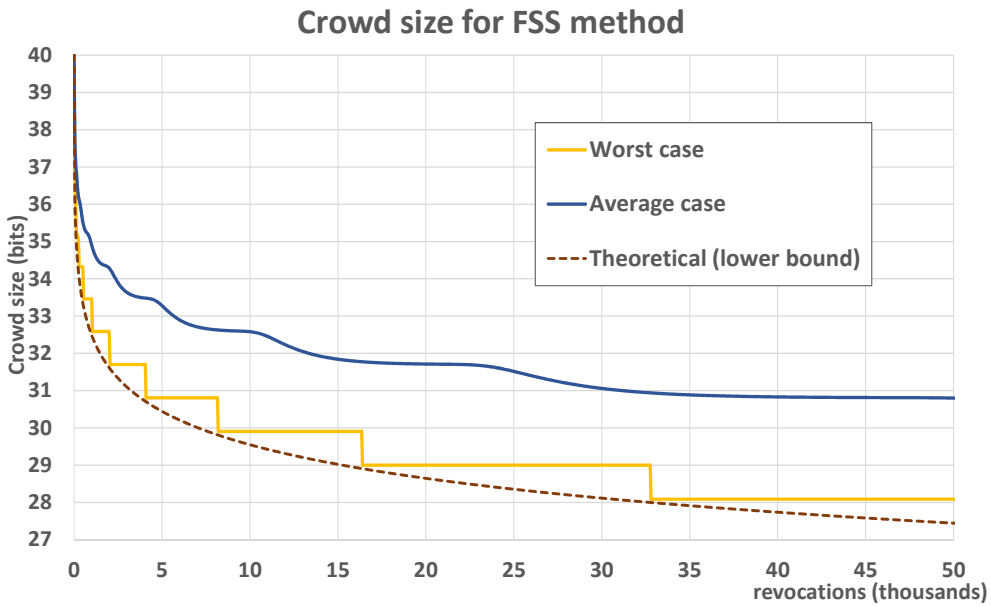
Source: the author.

96-byte message sent via a mutually-authenticated channel.

6.3 Crowd Size

Whereas the privacy obtained by the DR approach is potentially as absent as in C-ITS, and VSS provides a configurable privacy level, it is worth evaluating the crowd size obtained with the FSS approach. The results for $0 \leqslant n_r \leqslant 50{,}000$, considering our simulations and the lower bound formula given in Section 4.4.2, are shown in Figure 18.

Figure 18: Crowd size obtained with the FSS approach, for different numbers of revocations.



Source: the author.

This figure shows that, as expected, the crowd size obtained when requesting D nodes from the activation tree is inversely proportional to the number of revoked vehicles. Nevertheless, the average and worst-case privacy loss decrease roughly logarithmically with n_r. After all, to remove all nodes from the activation tree's depth $depth$, so they cannot be picked for increasing the crowd size by $2^{D-depth}$, there must be at least 2^{depth} revocations. Hence, nodes from the shallowest depths become unavailable quite quickly. Nevertheless, those from intermediate depths, which are still able to provide

a large amount of privacy when picked, require a huge number of revocations before they cannot be included in vehicles' requests anymore. Indeed, our simulations show that, for 50,000 revocations, on average a vehicle can still be confused with more than 2^{30} (out of 2^{40}) of its peers when requesting activation codes using the FSS strategy.

6.4 Summary

In this chapter, we compared the ACPC with other state-of-the-art alternative VPKIs. On the one hand, ACPC differs in two aspects when compared to IFAL: (1) while in IFAL a dishonest PCA could link several certificates to the same vehicle, in ACPC it would require a collusion between dishonest PCA and RA; and (2) ACPC distributes activation codes efficiently via broadcast, while in IFAL each vehicle must request them. On the other hand, although the security of BCAM is similar to ACPC, our proposed protocol is more efficient in terms of data bandwidth and processing time: it saves bandwidth in two interfaces, in the RA-CAM, depending on the batch size and number of periods per activation, and in the CAM-vehicle to activate, for the smaller code length; and it executes fewer operations than BCAM because it replaces an encryption by a modified operation in the butterfly key expansion. This gain actually depends on the system parameters, but ACPC provides better efficiency for larger batch sizes or longer activation periods. We also compared all the distribution methods (broadcast, DR, FSS and VSS) with other relevant revocation approaches, such as C-ITS, and the original SCMS with CRLs. We concluded by comparing the expected privacy achieved by FSS with varying the number of revoked vehicles in the system.

7 CONCLUSION

Security and privacy mechanisms are essential for preventing drivers from abusing V2X communications to gain unwarranted advantages over their peers, and also for limiting any entity's ability to track honest vehicles. A promising solution to address such requirements is the Security Credential Management System (SCMS), which provides efficient and privacy-preserving mechanisms for issuing pseudonym certificates to vehicles and revoking them in case of misbehavior. As a drawback, however, SCMS's revocation procedure is such that, after certificate identifiers are included in a CRL, it may take a long time for the corresponding CRL entries to be removed. This incurs not only in bandwidth overheads for the CRL distribution, but also increases the processing cost at vehicles for verifying a certificate's revocation status.

Aiming to address these issues, in this thesis we present improvements on SCMS's pseudonym certificate revocation process. The proposed design, named Activation Codes for Pseudonym Certificates (ACPC), is based on activation codes, small pieces of information without which pseudonym certificates previously issued become useless. Consequently, by ensuring that only non-revoked vehicles are periodically provided with those codes, entries associated with revoked vehicle's certificates can be safely removed from CRLs.

Even though ACPC builds upon ideas originally discussed in recent works, in particular IFAL (VERHEUL, 2016) and BCAM (KUMAR; PETIT; WHYTE, 2017), it leads to a significantly more secure and efficient pseudonym certificate system. More

precisely, SCMS's design is such that a dishonest PCA would have to collude with the RA to be able to track vehicles, and ACPC does not increase this vulnerability surface. In contrast, in IFAL a dishonest PCA could track vehicles without colluding with any other entities, whereas in BCAM a collusion between CAM and PCA would enable them to similarly violate the drivers' privacy, In terms of performance, one advantage of ACPC over IFAL refers to the distribution of activation codes: the CAM can broadcast codes to all non-revoked vehicles, so no bidirectional connectivity is required between vehicles and CAM. Even though such distribution process is as efficient as BCAM's, ACPC leads to a much more efficient certificate issuance process than BCAM: since the proposed design avoids the re-encryption of pseudonym certificates by the CAM, the CAM-RA bandwidth usage in ACPC is expected to be lower than 1/480 of BCAM's, and the processing overhead added by ACPC's activation codes at the vehicle's side is expected to be 1/80 or less than what is observed with BCAM.

We also propose alternative methods to its mechanism for distributing activation codes. The original distribution via broadcast preserves the anonymity of the receiver and does not require vehicles to have bi-directional connectivity. When requiring unicast communication, however, the bandwidth costs do not boil down to a single activation code, but grow approximately linearly with the number of revoked vehicles, reaching the order of megabytes for about 10,000 revocations. The goal of the hereby proposed methods is, thus, to strike a balance between privacy and bandwidth efficiency. In particular, in a scenario where vehicles can abdicate from anonymity when requesting/activating their certificates (e.g., similarly to C-ITS and IFAL), the direct request (DR) approach allows this task to be accomplished by means of a simple 16-byte activation code. Conversely, if a vehicle prefers to retain some level of anonymity when requesting activation codes, it can either: (1) download the entire activation tree, for maximum privacy but also maximum bandwidth usage; (2) employ the fixed-size

subset (FSS) method, downloading less than 1 KiB for a privacy level that degrades logarithmically with the number of revocations n_r; or (3) adopt the variable-size subset (VSS) approach, for which a configurable privacy level can be obtained with a logarithm increase on the download costs as n_r grows. Such gains were confirmed through simulations, and are particularly interesting when ACPC's activation tree encoding can be cached and simply updated whenever required.

7.1 Publications

As a direct result of this work, we produced the following publications:

- (SIMPLICIO et al., 2018a) is a journal article that present the ACPC, describing the security mechanisms to improve the revocation process of the SCMS promoting the distribution of activation codes by broadcast. The proposed solution to create activation trees was later incorporated in the IEEE standard 1609.2 for vehicular communication (IEEE, 2022).

- (SIMPLICIO et al., 2021b) is a journal article that discusses the bandwidth costs to distribute activation trees by unicast messages, and the proposed distribution strategies that balance privacy and efficiency.

Additionally, the author has collaborate in other publications, regarding V2X and other scenarios:

- (SIMPLICIO et al., 2018b) is a conference paper that improves the linkage procedure in SCMS in two fronts: firstly, it proposes a solution to a possible birthday attack that could be applied to SCMS if a large number of certificates are revoked; and secondly, it proposes an hierarchical way to create linkage values, so that SCMS can temporarily revoke vehicles, for example, to aid in investigations by law-enforcement authorities.

- (SIMPLICIO et al., 2018c) is a conference paper that improve efficiency in the generation of butterfly keys using an efficient mechanism that allows the vehicle to receive a single key from the RA that would allow it to both decrypt the received certificate and sign messages with no detriment to security due to correlation attacks.

- (SILVA et al., 2019) is a conference paper and demonstration of a library for secure drawing of juror or judges in legal systems. The library implements a method for the legal proceedings' stakeholders to contribute to the randomness of the drawing, so that the fairness of the selection could be audited. A full paper is still in development.

- (SILVA; SIMPLICIO, 2020) was a presentation of an extended abstract that proposes the usage of a tree-based key-agreement protocol for in-vehicle communication. It proposed three improvements to the original protocol: (1) a transformation from FFC to ECC to improve size and efficiency of communication; (2) a improvement to the node addition policy, so that in some situations the worst-case scenario could become the best-case; and (3) a proposal of node addition policy for temporary nodes (e.g., media devices, diagnostics tools). A full paper is still in development.

- (SIMPLICIO et al., 2021a) is a journal article that extends (SIMPLICIO et al., 2018b). It further improves the linkage procedure by proposing a way to simplify the SCMS architecture by not requiring the LAs to create linkage values for revoking certificates from the same vehicle, thus reducing deployment costs and the attack surface.

7.2 Future Work and Open Problems

In this thesis, we presented the ACPC scheme, which improves SCMS's revocation procedure by using activation codes, resulting in a more flexible way to revoke pseudonym certificates without much overhead when verifying CRLs. However, we believe that the area of V2X communications is still open to many improvements and proposals. Thus, we present here a few open problems related to ACPC that have not been solved yet.

SCMS formal proofs: Since ACPC is built upon SCMS, much of the original structure was reused in our proposal. Therefore, our security discussion is also based on the one from SCMS, in a way to argue that our enhancements do not hinder SCMS's security while providing the new functionality. As previously discussed in Chapter 5, the SCMS security analysis follows an informal approach (BRECHT et al., 2018) (WHYTE et al., 2013). Although no security flaws have been found so far, it could benefit from more formal proofs, such as the one based on the Game-Playing Technique (BELLARE; ROGAWAY, 2006), similar to what is proposed in IFAL (VERHEUL; HICKS; GARCIA, 2019). Hence, once formal proofs to SCMS are available, we could also adapt them to ACPC.

Post-quantum security: ACPC and SCMS were originally proposed using classical cryptographic primitives based on the ECDLP and parameterized to achieve a security level of 128 bits. However, when we consider the advances in quantum computing, building a quantum computer with enough resources will break classical asymmetric primitives, and symmetric primitives will require larger key sizes. For this reason, when reaching for long-term privacy, it may be appropriate to adapt the V2X schemes to work with post-quantum primitives. While qSCMS (BARRETO et al., 2018) proposes the usage of lattices to SCMS, we leave as an open problem its adaptation to ACPC.

REFERENCES

ADLEMAN, L. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *20th Annual Symposium on Foundations of Computer Science (SFCS 1979)*. San Juan, Puerto Rico: IEEE, 1979. p. 55–60. ISSN 0272-5428.

AGRAWAL, D.; ABBADI, A.; EMEKCI, F.; METWALLY, A. Database management as a service: Challenges and opportunities. In *IEEE 25th Int. Conf. on Data Engineering*. Shanghai, China: IEEE, 2009. p. 1709–1716. ISSN 1063-6382.

AIELLO, W.; LODHA, S.; OSTROVSKY, R. Fast digital identity revocation (extended abstract). In *Advances in Cryptology (CRYPTO 1998)*. London, UK: Springer, 1998. p. 137–152. ISBN 3-540-64892-5.

ARANHA, D.; GOUVÊA, C. *RELIC is an Efficient LIbrary for Cryptography*. 2018. <https://github.com/relic-toolkit/relic>.

BARKER, E. NIST special publication 800-57 part 1, revision 4. *NIST special publication*, vol. 800, no. 57, p. 1–147, 2016.

BARRETO, P. S.; RICARDINI, J. E.; JR, M. A. S.; PATIL, H. K. qscms: Post-quantum certificate provisioning process for v2x. *Cryptology ePrint Archive*, 2018.

BELLARE, M.; ROGAWAY, P. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - EUROCRYPT 2006*. Berlin, Heidelberg: Springe, 2006. p. 409–426. ISBN 978-3-540-34547-3. Available from Internet: <https://eprint.iacr.org/2004/331.pdf>.

BERNSTEIN, D. J. Curve25519: New Diffie-Hellman speed records. In *Public Key Cryptography - PKC 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 207–228. ISBN 978-3-540-33852-9.

BIHAM, E. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Information Processing Letters*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, vol. 84, no. 3, p. 117–124, 2002. ISSN 0020-0190.

BOEYEN, S.; SANTESSON, S.; POLK, T.; HOUSLEY, R.; FARRELL, S.; COOPER, D. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. [S.l.], 2008. 1-151 p. Available from Internet: <https://tools.ietf.org/html/rfc5280>.

BRECHT, B.; THERRIAULT, D.; WEIMERSKIRCH, A.; WHYTE, W.; KUMAR, V.; HEHN, T.; GOUDY, R. A security credential management system for V2X communications. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, vol. 19, no. 12, p. 3850–3871, Dec 2018. ISSN 1524-9050.

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>.

CHAUM, D. Blind signatures for untraceable payments. In SPRINGER. *Advances in cryptology*. [S.l.], 1983. p. 199–203.

COOPER, D. A more efficient use of delta-CRLs. In IEEE. *Proc. 2000 IEEE Symposium on Security and Privacy. S&P 2000*. [S.l.], 2000. p. 190–202.

DIMITRAKOPOULOS, G.; DEMESTICHAS, P. Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, IEEE, vol. 5, no. 1, p. 77–84, March 2010. ISSN 1556-6072.

DOUCEUR, J. The Sybil attack. In *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS 2002)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

ETSI. *ETSI TS 102 731 V1.1.1 – Intelligent Transport Systems (ITS); Security; Security Services and Architecture*. [S.l.], 2010.

ETSI. *EN 302 636-5-1 V1.1.1 – Intelligent Transport Systems (ITS); Communications Architecture*. [S.l.], 2017.

ETSI. *TS 103 097 V1.3.1 – Intelligent transport systems (ITS); security; Security header and certificate formats*. [S.l.], 2017.

ETSI. *TS 102 941 V2.1.1 – Intelligent Transport Systems (ITS); Security; Trust and Privacy Management*. [S.l.], 2021.

EU. *Certificate Policy for Deployment and Operation of European Cooperative Intelligent Transport Systems (C-ITS) Release 1.1*. [S.l.], 2018. Available from Internet: <https://ec.europa.eu/transport/sites/transport/files/c-its_certificate_policy-v1.1-track_changes.pdf>.

FIGUEIREDO, L.; JESUS, I.; MACHADO, J.; FERREIRA, J.; CARVALHO, J. Towards the development of intelligent transportation systems. In *Proceedings of the IEEE Intelligent Transportation Systems (ITSC 2001)*. Oakland, USA: IEEE, 2001. p. 1206–1211.

FÖRSTER, D.; KARGL, F.; LÖHR, H. PUCA: A pseudonym scheme with user-controlled anonymity for vehicular ad-hoc networks (VANET). In *IEEE Vehicular Networking Conference (VNC 2014)*. Paderborn, Germany: IEEE, 2014. p. 25–32. ISSN 2157-9857.

GÜNTHER, C. G. An identity-based key-exchange protocol. In SPRINGER. *Advances in Cryptology—EUROCRYPT'89: Workshop on the Theory and Application of Cryptographic Techniques Houthalen, Belgium, April 10–13, 1989 Proceedings 8*. [S.l.], 1990. p. 29–37.

GUO, J.; BAUGH, J. P.; WANG, S. A group signature based secure and privacy-preserving vehicular communication framework. In IEEE. *2007 Mobile Networking for Vehicular Environments*. Anchorage, USA: IEEE, 2007. p. 103–108.

HAAS, J.; HU, Y.; LABERTEAUX, K. Design and analysis of a lightweight certificate revocation mechanism for VANET. In ACM. *Proceedings of the 6th ACM International Workshop on VehiculAr InterNETworking (VANET 2009)*. Beijing, China: ACM, 2009. p. 89–98.

HAAS, J.; HU, Y.; LABERTEAUX, K. Efficient certificate revocation list organization and distribution. *IEEE Journal on Selected Areas in Communications*, IEEE, vol. 29, no. 3, p. 595–604, 2011. ISSN 0733-8716.

HANKERSON, D.; MENEZES, A. J.; VANSTONE, S. *Guide to Elliptic Curve Cryptography*. New York, USA: Springer, 2004. ISBN 0-387-95273-X.

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.

HUSEMÖLLER, D. *Elliptic curves*. 2. ed. New York, USA: Springer, 2004. ISBN 978-0-387-21577-8.

IEEE. IEEE standard specifications for public-key cryptography – amendment 1: Additional techniques. *IEEE Std 1363a-2004 (Amendment to IEEE Std 1363-2000)*, IEEE, p. 1–167, 2004.

IEEE. *IEEE Standard Specifications for Public-Key Cryptography – Amendment 1: Additional Techniques*. [S.l.], 2004.

IEEE. IEEE standard for wireless access in vehicular environments – security services for applications and management messages. *IEEE Std 1609.2*, IEEE, p. 1–229, 2016.

IEEE. *IEEE 1609.2.1-2022 IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Certificate Management Interfaces for End Entities*. New York, USA, 2022. Available from Internet: <https://standards.ieee.org/ieee/1609.2.1/10728/>. Cited 9/3/2022.

IYER, A.; KHERANI, A.; RAO, A.; KARNIK, A. Secure V2V communications: Performance impact of computational overheads. In *2008 IEEE INFOCOM Workshops*. Phoenix, USA: IEEE, 2008. p. 1–6.

JEONG, J. *IPv6 Wireless Access in Vehicular Environments (IPWAVE): Problem Statement and Use Cases (Internet-Draft)*. [S.l.], 2020. Available from Internet: <https://tools.ietf.org/id/draft-ietf-ipwave-vehicular-networking-19.html>.

KATZ, J.; MENEZES, A. J.; OORSCHOT, P. C. V.; VANSTONE, S. A. *Handbook of applied cryptography*. [S.l.]: CRC press, 1996.

KHODAEI, M.; PAPADIMITRATOS, P. The key to intelligent transportation: Identity and credential management in vehicular communication systems. *IEEE Vehicular Technology Magazine*, IEEE, vol. 10, no. 4, p. 63–69, Dec 2015. ISSN 1556-6072.

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 2017)*. New York, USA: ACM, 2017. p. 145–155. ISBN 978-1-4503-5084-6.

LAMPORT, L. Password authentication with insecure communication. *Magazine Communications of the ACM*, ACM, New York, USA, vol. 24, no. 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>. Cited 04/17/2019.

MICHALSKI, R.; VADEKAR., A. Opportunities for enhancing the robustness and functionality of the dedicated short range communications (DSRC) infrastructure through the use of satellite DARS to improve vehicle safety in the 21st century. In *34th AIAA International Communications Satellite Systems Conference (ICSSC)*. Cleveland, USA: American Institute of Aeronautics and Astronautics, 2013.

MILLER, V. Use of elliptic curves in cryptography. In SPRINGER. *Advances in Cryptology (CRYPTO 1985)*. Santa Barbara, USA: Springer Berlin Heidelberg, 1986. p. 417–426.

MOALLA, R.; LONC, B.; H.LABIOD; SIMONI, N. Risk analysis study of ITS communication architecture. In *3rd International Conference on The Network of the Future*. Gammarth, Tunisia: IEEE, 2012. p. 2036–2040.

NIST. *FIPS PUB 197 – Advanced Encryption Standard (AES)*. Gaithersburg, USA, 2001. Available from Internet: <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf>. Cited 01/02/2019.

NIST. *SP 800-38A – Recommendation for block cipher modes of operation: Methods and techniques*. Gaithersburg, USA, 2001. Available from Internet: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf>. Cited 04/17/2019.

NIST. *SP 800-38C – Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*. Gaithersburg, MD, USA, 2004.

NIST. *FIPS 198-1 – The Keyed-Hash Message Authentication Code (HMAC)*. Gaithersburg, MD, USA, 2008.

NIST. *SP 800-108 – Recommendation for Key Derivation Using Pseudorandom Functions*. Gaithersburg, USA, 2009. Available from Internet: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-108.pdf>. Cited 01/02/2019.

NIST. *FIPS PUB 186-4: Digital Signature Standard (DSS)*. Gaithersburg, USA, 2013. Available from Internet: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>. Cited 01/02/2019.

NIST. *FIPS PUB 180-4 – Secure Hash Standard (SHS)*. Gaithersburg, USA, 2015. Available from Internet: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>. Cited 01/02/2019.

NOWATKOWSKI, M. *Certificate Revocation List Distribution in Vehicular Ad Hoc Networks*. PhD Thesis (PhD) — Georgia Institute of Technology, Atlanta, GA, USA, 2010.

PAPADIMITRATOS, P.; FORTELLE, A. L.; EVENSSEN, K.; BRIGNOLO, R.; COSENZA, S. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *IEEE Communications Magazine*, IEEE, vol. 47, no. 11, p. 84–95, 2009. ISSN 0163-6804.

PORNIN, T. *Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)*. [S.l.], 2013. Available from Internet: <https://www.rfc-editor.org/rfc/rfc6979>.

PRENEEL, B. Encyclopedia of cryptography and security: Davies–Meyer hash function. In ____. Boston, MA: Springer, 2005. p. 136–136. ISBN 978-0-387-23483-0.

RAYA, M.; PAPADIMITRATOS, P.; AAD, I.; JUNGELS, D.; HUBAUX, J.-P. Eviction of misbehaving and faulty nodes in vehicular networks. *IEEE Journal on Selected Areas in Communications*, IEEE, vol. 25, no. 8, 2007.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, ACM, vol. 21, no. 2, p. 120–126, 1978.

SALEM, F. M.; IBRAHIM, M. H.; IBRAHIM, I. Non-interactive authentication scheme providing privacy among drivers in vehicle-to-vehicle networks. In IEEE. *2010 Sixth International Conference on Networking and Services (ICNS)*. Cancun, Mexico: IEEE, 2010. p. 156–161.

SANTESSON, S.; MYERS, M.; ANKNEY, R.; MALPANI, A.; GALPERIN, S.; ADAMS, C. *RFC 6960: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*. [S.l.], 2013.

SCHAUB, F.; MA, Z.; KARGL, F. Privacy requirements in vehicular communication systems. In *Proceedings of the International Conference on Computational Science and Engineering*. Vancouver, Canada: IEEE, 2009. vol. 3, p. 139–145.

SHAMIR, A. How to share a secret. *Communications of the ACM*, ACM, New York, USA, vol. 22, no. 11, p. 612–613, 1979. ISSN 0001-0782.

SHIREY, R. W. *Internet Security Glossary, Version 2*. [S.l.], 2007. 1-365 p. Available from Internet: <https://tools.ietf.org/html/rfc4949>. Cited 04/17/2019.

SHOUP, V. *A computational introduction to number theory and algebra*. [S.l.]: Cambridge University Press, 2009.

SILVA, M. V. M.; SIMPLICIO, M. A. Library application for a fair, traceable, auditable and participatory drawing tool for legal systems. In *Extended Proceedings of the XX Brazilian Symposium on Information and Computational Systems Security*. Porto Alegre, RS, Brazil: SBC, 2020. p. 117–124. Available from Internet: <https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/19278>.

SILVA, M. V. M.; SIMPLICO, M. A.; COMINETTI, E. L.; RICARDINI, J. E.; OGAWA, H.; CUNHA, H.; PATIL, H. K. *Optimizing Group Key-agreement in Automotive In-Vehicle Networks*. 2019. 9th ESCAR USA (In-Person Event).

SILVERMAN, J. H.; SUZUKI, J. Elliptic curve discrete logarithms and the index calculus. In SPRINGER. *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 1998)*. Beijing, China: Springer Berlin Heidelberg, 1998. p. 110–125.

SIMPLICIO, M.; SILVA, M.; COMINETTI, E. *ACPC Tree Simulator*. 2019. Code Ocean: <https://doi.org/10.24433/CO.7991543.v1>.

SIMPLICIO, M. A.; COMINETTI, E. L.; PATIL, H. K.; RICARDINI, J. E.; SILVA, M. V. M. ACPC: Efficient revocation of pseudonym certificates using activation codes. *Ad Hoc Networks*, Elsevier, 2018. ISSN 1570-8705. See also: <https://eprint.iacr.org/2018/324>.

SIMPLICIO, M. A.; COMINETTI, E. L.; PATIL, H. K.; RICARDINI, J. E.; FERRAZ, L. T. D.; SILVA, M. V. M. A privacy-preserving method for temporarily linking/revoking pseudonym certificates in VANETs. In *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom 2018)*. New York, USA: IEEE, 2018. See also <https://eprint.iacr.org/2018/185>.

SIMPLICIO, M. A.; COMINETTI, E. L.; PATIL, H. K.; RICARDINI, J. E.; SILVA, M. V. M. The unified butterfly effect: Efficient security credential management system for vehicular communications. In *2018 IEEE Vehicular Networking Conference (VNC 2018)*. Taipei, Taiwan: IEEE, 2018. See also: <https://eprint.iacr.org/2018/089>.

SIMPLICIO, M. A.; COMINETTI, E. L.; PATIL, H. K.; RICARDINI, J. E.; FERRAZ, L. T. D.; SILVA, M. V. M. Privacy-preserving certificate linkage/revocation in vanets without linkage authorities. *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, p. 3326–3336, 2021.

SIMPLICIO, M. A.; COMINETTI, E. L.; PATIL, H. K.; RICARDINI, J. E.; SILVA, M. V. M. Revocation in vehicular public key infrastructures: Balancing privacy and efficiency. *Vehicular Communications*, Elsevier, vol. 28, p. 100309, 2021.

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>.

VERHEUL, E.; HICKS, C.; GARCIA, F. D. Ifal: Issue first activate later certificates for v2x. In IEEE. *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. [S.l.], 2019. p. 279–293.

WEIL, A. L'arithmétique sur les courbes algébriques. *Thèses françaises de l'entre-deux-guerres*, Kluwer Academic Publishers, vol. 95, p. 1–35, 1928.

WHYTE, W.; WEIMERSKIRCH, A.; KUMAR, V.; HEHN, T. A security credential management system for V2V communications. In *IEEE Vehicular Networking Conference (VNC 2013)*. Boston, USA: IEEE, 2013. p. 1–8. ISSN 2157-9857.

YU, W.; BAI, W.; QI, L.; LUAN, W. State-of-the-art review on traffic control strategies for emergency vehicles. *IEEE Access*, IEEE, 2022.

APPENDIX A – BIRTHDAY ATTACK AGAINST BCAM'S HASH TREES

The structure of BCAM's binary hash trees is such that their k-bit nodes are computed via iterative hashing, using a constant suffix for each branch. More precisely, starting from a random root $\mathtt{node}_t(0,0)$, each node $\mathtt{node}_t(depth, count)$ of $\mathtt{tree}_t$ is computed from its parent as follows:

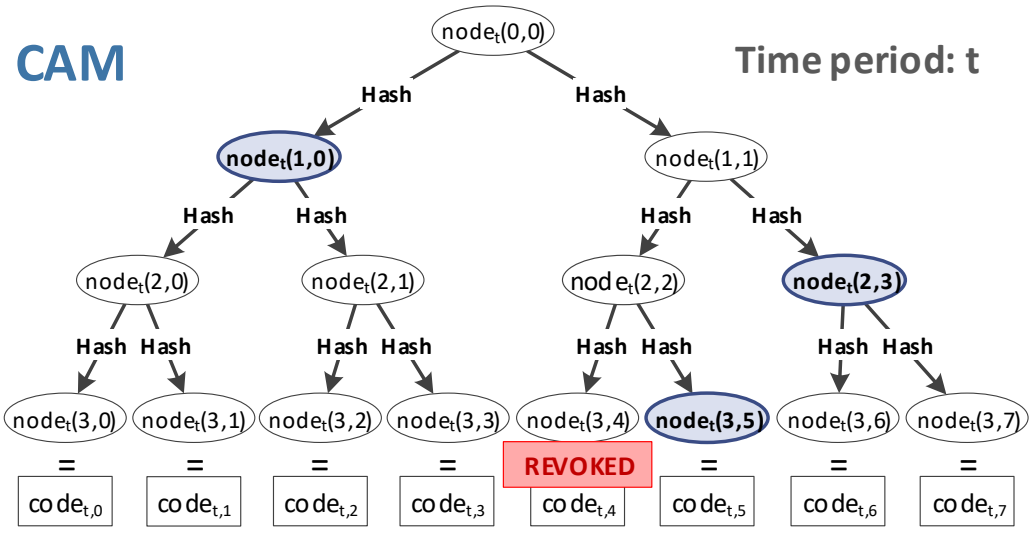
$$\mathtt{node}_t(depth, count) = Hash(\mathtt{node}_t(depth-1, \lfloor count/2 \rfloor) \,\|\, b^p),$$

where $b = 0$ (resp. $b = 1$) if the node is a left (resp. right) child, and $p \geqslant 1$ is a suitable padding length. For example, when $k = 256$ and the hash function employed is SHA-256, adopting $1 \leqslant p < 192$ would allow the underlying compression function to be called only once when computing any node of the tree.

Suppose that a vehicle with identifier VID_r is revoked. In that case, the leaf $\mathtt{node}_t(|\mathrm{VID}|, \mathrm{VID}_r)$ should not be computed from the message broadcast by the CAM, for every future value of t. This means that the set N_r of all nodes in the path between the root and that leaf must remain secret. To accomplish this, the CAM only broadcasts *siblings* of the nodes in N_r. For example, as mentioned in Section 4.3 (and replicated in Figure 19), the revocation of $\mathtt{node}_t(3,4)$ leads to the disclosure of the set $N_d = \{\mathtt{node}_t(1,0), \mathtt{node}_t(2,3), \mathtt{node}_t(3,5)\}$. As long as the tree is built using a secure hash function, it is not straightforward to use any node in N_d to compute nodes in the set

$N_r = \{\mathsf{node}_t(0,0), \mathsf{node}_t(1,1), \mathsf{node}_t(2,2), \mathsf{node}_t(3,4)\}$. Indeed, doing so corresponds to finding preimages for nodes in the set N_d.

Figure 19: Distribution of activation tree nodes when vehicle with $VID = 4$ is revoked.



Source: the author

To overcome the security of BCAM's activation trees, the following attack strategy can be employed to recover activation codes for revoked vehicles. First, the attacker picks an arbitrary k-bit long $\mathtt{link}_0$, and arbitrarily chooses between $b = 0$ or $b = 1$. The value of $\mathtt{link}_0$ is then used as the anchor for a hash chain of the form $\mathtt{link}_j = \mathit{Hash}(\mathtt{link}_{j-1} \| b^p)$, until 2^n hashes are performed. For simplicity, we assume that no collision occurs during this process, i.e., that $\mathtt{link}_j \neq \mathtt{link}_{j'}$ for all $j \neq j'$. Nevertheless, this simplification comes without loss of generality because, whenever there is a collision, the attacker could simply (1) save the current chain, (2) pick a new anchor distinct from any previously computed $\mathtt{link}_j$, and then (3) start a new chain from this anchor. Actually, picking different anchors for building multiple chains is likely advantageous anyway, because this facilitates the parallel processing of hashes. As long as 2^n different hashes are made available in this manner, the attack can proceed.

Due to the birthday paradox, an attacker that gathers 2^m nodes disclosed by the

CAM has a high probability to find a match between at least one of those nodes and some of the 2^n previously computed $\texttt{link}_j$ if $(m+n \geqslant k)$. Suppose that a match occurs between $\texttt{link}_j$ and $\texttt{node}_t(depth, count)$. In this case, $\texttt{link}_{j-1}$ is a valid preimage for $\texttt{node}_t(depth, count)$ with padding b^p. Hence, if the attacker picked $b = 0$ and $\texttt{node}_t(depth, count)$ is a left child, it is very likely that $\texttt{link}_{j-1}$ will match the parent of $\texttt{node}_t(depth, count)$ in the activation tree — unless $\texttt{link}_{j-1}$ is a second preimage rather than the actual preimage. If the parent of $\texttt{node}_t(depth, count)$ is also a left child, its own parent is also likely to match $\texttt{link}_{j-2}$, and so forth. An analogous argument applies if $b = 1$ and $\texttt{node}_t(depth, count)$ is a right child. As a result, such collisions have roughly 50% of chance of giving the attacker access to nodes belonging to the revoked set N_r. All certificates whose revocation depended on those nodes can then be activated.

Considering this attack scenario, the growth of the number of revoked devices has two negative effects on the system's security. First, the recovery of one node from the set N_r becomes more likely to give access to activation codes of multiple revoked devices. The reason is that a node in a given position of the tree always allows the computation of a same number of leafs (the lower the depth, the higher this number). When the number of revoked devices increase, so does the number of leaves covered by that node that should remain concealed to prevent the corresponding activation codes from being recovered. Second, the number of nodes disclosed by the CAM that would lead to useful collisions also grows, i.e., the value of m becomes larger.

Since such attacks trade time for space, one possible defense strategy is to adopt a large enough k parameter. For example, the authors of BCAM suggest $k = 256$ (cf. (KUMAR; PETIT; WHYTE, 2017), Section 4.1.3), meaning that the attacker would have to compute, say, $2^n = 2^{128}$ hashes and then gather $2^m = 2^{128}$ nodes from the CAM before a collision actually occurs. Therefore, in practice, the attacks hereby described do not pose an actual security threat to BCAM.

Nevertheless, there is a more efficient defense strategy for this issue, originally discussed by Leighton and Micali (LEIGHTON; MICALI, 1995) in the context of hash-based signatures (MCGREW; CURCIO; FLUHRER, 2017) and also proposed for use with SCMS's linkage trees (SIMPLICIO et al., 2018b): to use a different suffix for each node computation. This strategy comes from the observation that collisions between $\texttt{link}_j$ and $\texttt{node}_t(depth, count)$ are useless if they are computed with different suffixes. After all, in that case $\texttt{link}_{j-1}$ will not match the parent of $\texttt{node}_t(depth, count)$, i.e., it will necessarily be a second preimage rather than the actual preimage of that node. At the same time, attackers are unable to gather more than 1 value of $\texttt{node}_t(depth, count)$ for any given suffix. Consequently, to obtain a high probability of collisions for that suffix, the attacker would have to build a table with $2^n = 2^{k-m} = 2^k$ entries. In other words, the approach adopted in ACPC leads to a 128-bit security level even when the nodes themselves are 128-bit long.

APPENDIX B – C-ITS TICKET ISSUANCE MESSAGES

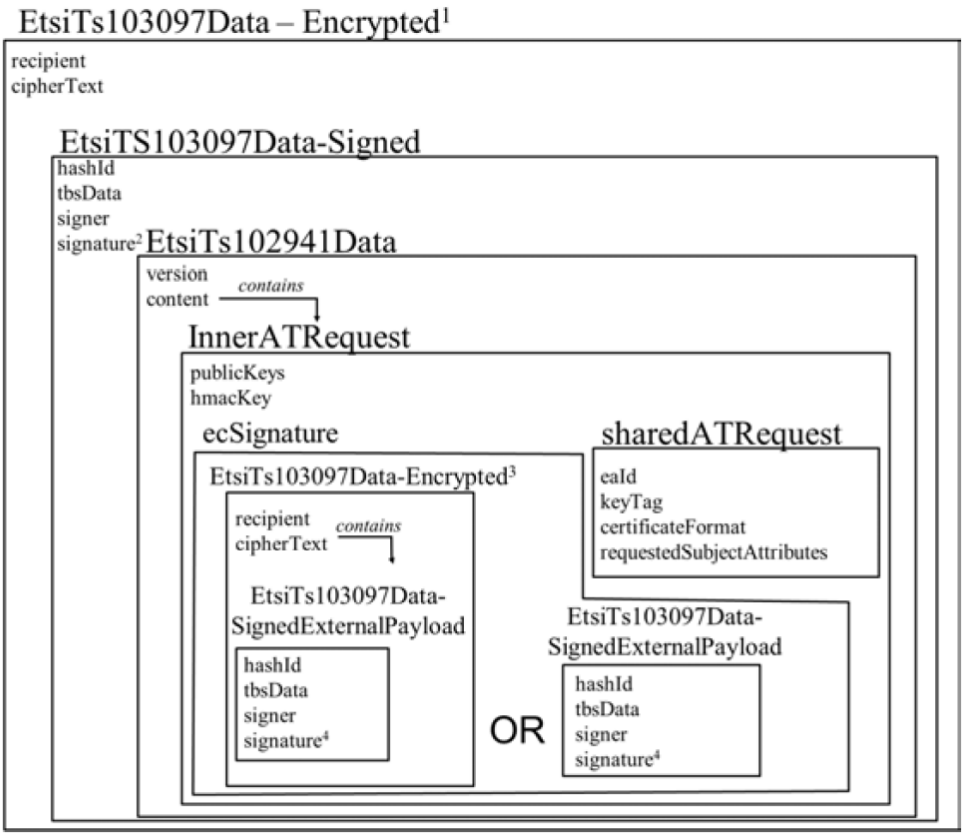
In C-ITS, vehicles must sent a Ticket Request (TReq) message to an Authorization Authority (AA) for each pseudonym certificate it needs to be issued. The AA then replies with a Ticket Response (TResp) message that encapsulates that certificate in a privacy-preserving manner. In this appendix, we describe the structure of both TReq and TResp aiming to estimate their sizes and, thus, support the comparisons made in Section 6.2.2. Namely, our estimate is that each TReq and TResp messages comprise, respectively, 505 and 265 bytes. This analysis complements the description of TReq and TResp data fields given in (ETSI, 2021) (ETSI, 2017b), where the actual sizes of such messages are not explicitly calculated.

B.1 C-ITS Ticket Request (TReq)

TReq's data structure is depicted in Figure 20, considering the version of the message that includes an inner Proof Of Possession (POP) signature. We note that, even though some instances of TReq may get smaller by omitting this signature, its presence is recommended in the official documentation (ETSI, 2021, Sec. 6.2.3.3.1), since it ensures the requester knows the private key corresponding to the requested certificate's public key. For convenience, in what follows we describe the message's structure from the inner- to the outermost field.

EtsiTs103097Data-SignedExternalPayload (115 bytes):

Figure 20: C-ITS Ticket Request (TReq) message structure.



NOTE: [1] Encryption is done with ECIES using the public encryption key of the AA.
[2] Signature computed using the private key corresponding to the verification public key to be certified.
[3] Encryption is done with ECIES using the public encryption key of the EA.
[4] Signature computed using currently valid EC private key corresponding to the verification public key.

Source: ETSI (2021, Fig. 17)

- **hashId** (1 byte): Indicates the hash algorithm to be used.
- **tbsData** (42 bytes): Contains the hash of the **sharedATRequest** (32 bytes), the provider service identifier (psid – 2 bytes), and the generation time (8 bytes).
- **signer** (8 bytes): References the vehicle's enrollment certificate by applying a hash function to it and using 8 bytes of the output.
- **signature** (64 bytes): Signature of **tbsData** using the vehicle's enrollment certificate.

EtsiTs103097Data-Encrypted (217 bytes):

- **recipient** (86 bytes): References the Enrollment Authority to be contacted; this is done by applying a hash function to the Enrollment Authority's certificate and using 8 bytes of the output. It also contains the encryption key material to be used the Enrollment Authority when encrypting the payload, using ECIES (IEEE, 2004b) and AES-CCM (NIST, 2004). The total space occupied includes the aforementioned 8-bytes hash value, the ECIES nonce, which is an elliptic curve point (33 bytes), the AES-CCM key encrypted and authenticated with ECIES (32 bytes), and the AES-CCM nonce and parameter (13 bytes).

- **ciphertext** (131 bytes): the AES-CCM encryption of the **EtsiTs103097Data -SignedExternalPayload** (115 bytes) and its authentication tag (16 bytes).

sharedATRequest (41 bytes):

- **eaId** (8 bytes): Identifies the Enrollment Authority to be contacted.

- **keyTag** (16 bytes): Leftmost 16 bytes of the hash of **hmacKey** concatenated with **publicKeys**.

- **certificateFormat** (4 bytes): The version of the certificate format specification (integer value set to 1).

- **requestedSubjectAttributes** (13 bytes): The attributes of the pseudonym certificate to be created. Namely: id, which may be set to null (1 byte); validity period (start and duration – 6 bytes); region (country and location – 3 bytes); assuranceLevel (1 byte); and appPermissions (sequence that represents the services of the psid – 2 bytes).

InnerATRequest (323 bytes):

- **publicKeys** (33 bytes): The public key enclosed in the pseudonym certificate, which corresponds to an elliptic curve point.

- **hmacKey** (32 bytes): A key to be used by the HMAC (NIST, 2008) function for generating **keyTag**.
- **sharedATRequest** (41 bytes) and **EtsiTs103097Data-Encrypted** (217 bytes): previously described.

EtsiTs102941Data (327 bytes):

- **version** (4 bytes): An integer value, set to 1.
- **content** (323 bytes): **InnerATRequest**.

EtsiTs102941Data-Signed (403 bytes):

- **hashId** (1 byte): Indicates the hash algorithm employed.
- **tbsData** (337 bytes): Contains the package payload, **EtsiTs102941Data** (327 bytes), a psid (2 bytes) and the generation time (8 bytes).
- **signer** (1 byte): May be set to 'self'.
- **signature** (64 bytes): Signature of **tbsData**. It is created using the private key whose corresponding public key is enclosed in the pseudonym certificate.

EtsiTs103097Data-Encrypted (505 bytes):

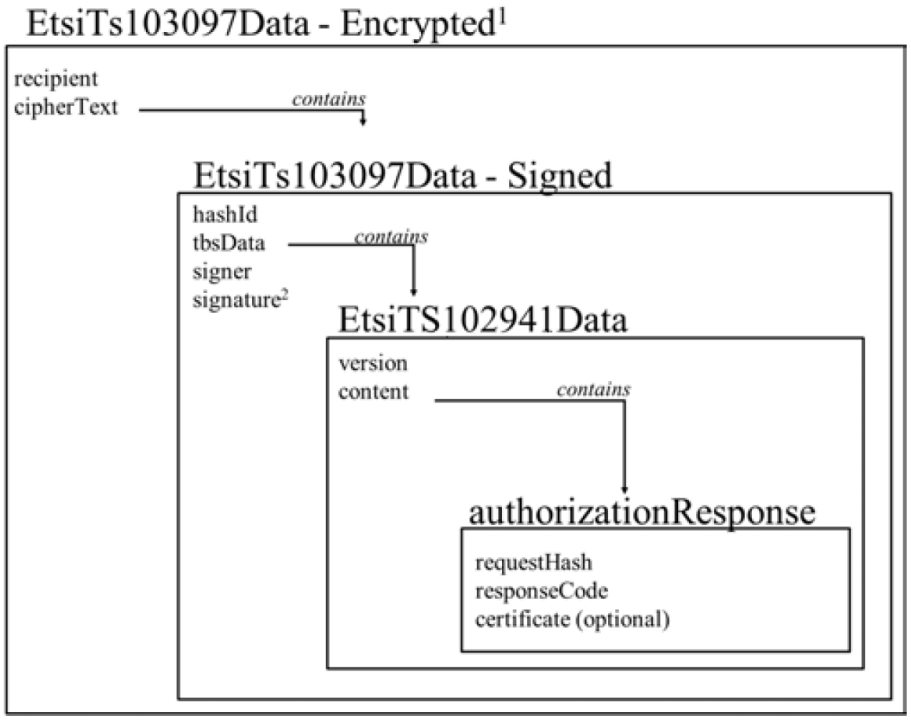
- **recipient** (86 bytes): References the AA to be contacted; this is done by applying a hash function to the AA's certificate and using 8 bytes of the output. It also contains the encryption key material to be used the AA when encrypting the payload, using ECIES (IEEE, 2004b) and AES-CCM (NIST, 2004). The total space occupied includes the AA reference (8 bytes), the ECIES nonce, which is an elliptic curve point (33 bytes), the AES-CCM key encrypted and authenticated with ECIES (32 bytes), and the AES-CCM nonce and parameter (13 bytes).

- **ciphertext** (419 bytes): the AES-CCM encryption of the **EtsiTs102941Data - Signed** (403 bytes) and its authentication tag (16 bytes).

B.2 C-ITS Ticket Response (TResp)

TResp's data structure is depicted in Figure 21. Similarly to TReq, in what follows we describe TResp from the inner- to the outermost field.

Figure 21: C-ITS Ticket Response (TResp) message structure.



NOTE: [1] Encryption is done with the AES key used for the encryption of the ATRequest.
[2] Signature computed using the verification private key associated with the AA certificate.

Source: ETSI (2021, Fig. 19)

authorizationResponse (154 bytes):

- **requestHash** (16 bytes): The 16 leftmost bytes of the hash computed over the corresponding TReq's **EtsiTs102941Data-Signed** field.
- **responseCode** (2 bytes): The code that indicates the result of the request (set to 0 if successful).

- **certificate** (136 bytes): The pseudonym certificate, comprising:
  - **version** (1 byte): Version of the certificate.
  - **type** (1 byte): Indicates whether this is an implicit or explicit certificate.
  - **issuer** (8 bytes): Identifies the issuer of the certificate.
  - **toBeSigned** (62 bytes): The core of the certificate. It contains: the certificate id, which may be set to 'NULL' (1 byte); the Certificate Revocation Authorizing Certificate Authority id (cracaId – 3 bytes); the validity period (start and duration – 6 bytes); the CRLseries (2 bytes); the region (country and location – 3 bytes); assuranceLevel (1 byte); appPermissions, the sequence that represents the services of the psid (2 bytes); the certificate request permissions, describing the permissions that apply to the certificate holder (11 bytes); and the certificate key (an elliptic curve point – 33 bytes).
  - **signature** (64 bytes): The AA's signature on the pseudonym certificate.

EtsiTS102941Data (158 bytes):

- **version** (4 bytes): An integer value, set to 1.
- **content** (154 bytes): The **authorizationResponse**.

EtsiTS103097Data-Signed (241 bytes):

- **hashId** (1 byte): Indicates the hash algorithm to be used.
- **tbsData** (168 bytes): The package's payload. It comprises: **EtsiTs102941Data** (158 bytes); a psid (2 bytes), set to "secured certificate request"; and the generation time (8 bytes).
- **signer** (8 bytes): The identifier of the AA responsible for signing the data.
- **signature** (64 bytes): Signature of **tbsData**, created by the AA.

EtsiTS103097Data-Encrypted (265 bytes):

- **recipient** (8 bytes): The hash of the symmetric key used in the corresponding TReq. It allows the vehicle to identify which key to use for decrypting this TResp.
- **ciphertext** (257 bytes): The AES-CCM encryption of the **EtsiTS103097Data -Signed** (241 bytes) and its MAC tag (16 bytes).