Schnorr-based implicit certification: improving the security and efficiency of V2X communications

Paulo S. L. M. Barreto[1], Marcos A. Simplicio Jr.[2], Jefferson E. Ricardini[2,3] and Harsh Kupwade Patil[3]

[1] University of Washington Tacoma, USA,
pbarreto@uw.edu,
[2] Escola Politécnica, Universidade de São Paulo, Brazil,
{mjunior,joliveira}@larc.usp.br
[3] LG Electronics, USA,
harsh.patil@lge.com

Abstract. In the implicit certification model, the process of verifying the validity of the signer's public key is combined with the verification of the signature itself. When compared to traditional, explicit certificates, the main advantage of the implicit approach lies in the shorter public key validation data. This property is particularly important in resource-constrained scenarios where public key validation is performed very often, which is common in vehicular communications (V2X) that employ pseudonym certificates. In this article, we show that an alternative, Schnorr-based implicit certification procedure can improve the efficiency of a popular V2X-oriented pseudonym certificate provisioning approach, the (unified) butterfly key expansion. As an additional contribution, we show that butterfly keys are vulnerable to existential forgery attacks under certain conditions, and also discuss how this issue can be fixed in an effective and efficient manner.

Keywords: Vehicular communications (V2X) · implicit certificates · butterfly key expansion · security

1 Introduction

Public key authentication is the cornerstone for any security system that relies on public/private key pairs for digital signature, key exchange and/or asymmetric encryption. Traditionally, this is accomplished by means of explicit certificates, digital documents that enclose the subject's public key and are signed by a trusted Certificate Authority (CA). This is the case of the worldwide Internet, which relies basically on X.509 certificates for verifying the authenticity of web domains [1].

Despite the widespread adoption of explicit certificates, there are some alternative models in the literature that target specific scenarios (see [2] for some examples). Among them, implicit certificates [3, 4] are of especial interest because, when compared to explicit certificates, they reduce the amount of information required for authenticating public keys. This better efficiency makes implicit certificates quite appealing in resource-constrained scenarios, such as the Internet of Things (IoT) [5, 6]. In particular, this is the certification model recommended in one of the main security architectures for protecting vehicular communications (also known as "vehicle-to-everything", or V2X): the Security Credential Management System (SCMS) [7, 8]. For example, by adoption the Elliptic Curve Qu-

Table 1: General notation and symbols

Symbol	Meaning
r	A random integer
sig	A digital signature
$cert$	A digital certificate
$\texttt{meta}$	A digital certificate's metadata
$\mathrm{U}, \mathcal{U}$	Public signature keys (stylized $\mathcal{U}$: reserved for PCA)
u, u	Private keys associated to U and $\mathcal{U}$ (respectively)
X, x	Public and private caterpillar keys
$\hat{X}, \hat{x}$	Public and private cocoon keys
V	Public key reconstruction credential (implicit model)
β	Number of cocoon keys in a batch of certificates
f	A pseudorandom function (PRF)
$\mathcal{H}(str)$	Hash of bitstring str
$Enc(\mathcal{K}, str)$	Encryption of bitstring str with key $\mathcal{K}$

Vanstone (ECQV) implicit certification model, SCMS creates certificates as small as 117 bytes [9].

In this article, we show that a more efficient public key verification process can be obtained if SCMS adopts an alternate implicit certification scheme. The proposed solution builds upon the Optimal Mail Certificate (OMC) implicit certification scheme [10], which enables optimization opportunities not available in ECQV. The result is that integrating the proposed scheme with SCMS's (unified) butterfly key expansion procedure [7, 8] leads to interesting efficiency gains on the vehicles' side. At the same time, our proposal addresses potential insecurity issues that might occur in this scenario. In particular, it prevents forgeries against OMC and similar designs, like the attack described in [11], as well as a novel existential forgery attack that applies specifically to butterfly keys.

The rest of this document is organized as follows. Section 2 summarizes the notation employed along this work. Section 3 briefly describes the goals and characteristics of regular implicit certificates. Section 4 introduces the proposed OMC-based implicit certification scheme and discusses its security and performance. Section 5 shows how the proposed approach can be integrated with SCMS and the corresponding performance gains; it also discusses existential forgery attacks against SCMS and how it is seamlessly averted in our proposal. Finally, Section 6 concludes the discussion.

2 Basic notations and definitions

Along this document, we write G to denote the generator of an elliptic curve group $\mathbb{G}$ of prime order q. All operations in this group are made $(\bmod q)$; hence, for conciseness, we omit this modular reduction when describing existing and the hereby proposed protocols.

For convenience of the reader, Table 1 lists the main symbols employed in the cryptographic protocols hereby described. In addition to them, we write $v \overset{\$}{\leftarrow} \mathbb{Z}_q$ to denote that v is randomly sampled from $\mathbb{Z}_q$, the finite field of integers modulo q. We also denote by $a \overset{?}{=} b$ the procedure of verifying whether or not a and b have identical values.

Table 2: ECQV implicit certificate provisioning protocol

usr	$\rightarrow$	CA	$\rightarrow$	usr
$x \xleftarrow{\$} \mathbb{Z}_q$ $X \leftarrow x{\cdot}G$	X	$r \xleftarrow{\$} \mathbb{Z}_q$ $V \leftarrow X + r{\cdot}G$ $cert \leftarrow \{V, \texttt{meta}\}$ $h \leftarrow \mathcal{H}(cert)$ $sig \leftarrow h{\cdot}r + u$	V, sig	$h \leftarrow \mathcal{H}(cert)$ $\mathrm{u} \leftarrow h{\cdot}x + sig$ $\mathrm{U} \leftarrow \mathrm{u}{\cdot}G$ $\mathrm{U} \overset{?}{=} h \cdot V + \mathcal{U}$

3 Implicit certification

The basic goal of digital certification is to bind public keys to their legitimate owners. In the traditional, explicit certification model, this is accomplished via a two-phase process. First, the user picks its private key u and then computes the corresponding public key $\mathrm{U} \leftarrow \mathrm{u}{\cdot}G$, independently of any other entity. Then, a trusted Certification Authority (CA) generates a digital certificate containing, at least, some system-defined metadata `meta`, the public key U, and the CA's signature on this data *sig*. A valid signature *sig* on certificate $(\texttt{meta}, \mathrm{U}, sig)$ usually implies that the owner of the certificate knows the private key u corresponding to U. Nonetheless, this knowledge can only be confirmed when U is actually used (e.g., by running a challenge-response protocol involving that key pair).

Conversely, in the implicit certification model the key pair (u, U) is computed in collaboration with the CA. What is provisioned by the CA in this case is not a certificate, but a public key reconstruction credential V bound to that user. By using V together with the CA's public key $\mathcal{U}$ and the metadata `meta`, any entity can obtain the user's corresponding public key U. Even though the reconstructed U is not explicitly signed by the CA, its authenticity and its bond to a user can still be proved. This is done by U's rightful owner simply by proving the knowledge of the private counterpart u, during key usage.

To give a concrete example, we can consider what is probably the main implicit certification solution in the literature, the Elliptic Curve Qu-Vanstone (ECQV) protocol [4]. ECQV's key provisioning procedure is shown in Table 2. First, the user computes a random EC private/public key pair $(x, X = x{\cdot}G)$, and then sends the public key X to the CA. The CA, in turn, picks a random integer r and computes a key reconstruction credential $V \leftarrow X + r{\cdot}G$. The CA then sets the user's implicit certificate to $cert \leftarrow \{V, \texttt{meta}\}$ and signs the result, obtaining $sig \leftarrow h{\cdot}r + u$ for $h \leftarrow \mathcal{H}(cert)$. Both *cert* and *sig* are then sent back to the requesting user. Finally, the user computes its own private key as $\mathrm{u} \leftarrow h{\cdot}x + sig$, and the corresponding public key as $\mathrm{U} \leftarrow \mathrm{u}{\cdot}G$.

In possession of *cert*, any user can reconstruct the public key U by computing $h \leftarrow \mathcal{H}(cert)$ and then using the equation $\mathrm{U} \leftarrow h{\cdot}V + \mathcal{U}$. This equation is valid because $\mathrm{U} = \mathrm{u}{\cdot}G = (h{\cdot}x + sig){\cdot}G = (h{\cdot}x + h{\cdot}r + u){\cdot}G = h{\cdot}(x{\cdot}G + r{\cdot}G) + u{\cdot}G = h{\cdot}V + \mathcal{U}$. As discussed in [3], this procedure prevents forgery of a valid key reconstruction credential V by attackers that do not know the CA's private key $\mathcal{U}$ used to sign *cert*. The scheme is also escrowless, i.e., the CA does not learn the user's final private key u.

Commonly, the metadata enclosed in *cert* includes some user-identifiable information [3, 4], similarly to the "subject" field in ordinary X.509 certificates [1]. Nevertheless, by omitting such identity information, one can obtain anonymous (or, rather, pseudonym) implicit certificates. In that case, the bearer of the certificate, after proving knowledge of the corresponding private key, is considered authorized in the system despite not being identified.

Table 3: Schnorr-based implicit certificate (SIMPL) provisioning protocol

usr	$\to$	CA	$\to$	usr
		$r \xleftarrow{\$} \mathbb{Z}_q$		$h \leftarrow \mathcal{H}(cert,\mathcal{U})$
$x \xleftarrow{\$} \mathbb{Z}_q$		$V \leftarrow X + r\cdot G$		$\mathrm{u} \leftarrow x + sig$
$X \leftarrow x\cdot G$	X	$cert \leftarrow \{V, \texttt{meta}\}$	V, sig	$\mathrm{U} \leftarrow \mathrm{u}\cdot G$
		$h \leftarrow \mathcal{H}(cert,\mathcal{U})$		$\mathrm{U} \overset{?}{=} V + h\cdot\mathcal{U}$
		$sig \leftarrow r + h\cdot u$		

4 More efficient implicit certificates

The main performance shortcoming of the ECQV protocol shown in Section 3 lies in its credential verification equation, $\mathrm{U} \overset{?}{=} h\cdot V + \mathcal{U}$. Specifically, the processing cost of this equation is dominated by the EC-point multiplication $h\cdot V$, where the V point is expected to be different for each user. Therefore, this operation is not amenable to optimization methods typical of fixed-point EC multiplications (see [12, Sec. 3.3.2] for a few examples). In other words, for a generic choice of the underlying elliptic curve, it cannot be accelerated by pre-computing and storing data that depend only on the point V.

To address this issue, one can adopt a slightly different implicit certification protocol that takes full advantage of fixed-point multiplications. The proposed solution is shown in Table 3, in which we highlight (in gray background) the differences toward ECQV. The first distinction is that, like the Optimal Mail Certificate (OMC) [10], we employ the elliptic curve version of the Schnorr signature algorithm [13] when computing $sig = r + h\cdot u$. The private key reconstruction operation then becomes $\mathrm{u} \leftarrow x + sig$, whereas the public key reconstruction algorithm is given by $\mathrm{U} = V + h\cdot\mathcal{U}$. The correctness of these equations can be verified by observing that $\mathrm{U} = \mathrm{u}\cdot G = (x+sig)\cdot G = (x+r+h\cdot u)\cdot G = (x\cdot G + r\cdot G) + h\cdot(u\cdot G) = V + h\mathcal{U}$.

The second difference is that, similarly to [14], the proposal includes the intended verification public key in the hash computation, by making $h = \mathcal{H}(cert,\mathcal{U})$. This approach is recommended for preventing certificate misbinding attacks [15, 16], in which the wrong CA public key is associated with the implicit credential. In particular, this prevents the following attack against the system's consistency. Suppose that adversary $\mathcal{A}$ acts as an intermediary in the communication between the CA and the user *usr* requesting an implicit certificate. Suppose also that *usr* in principle does not know the CA's public key $\mathcal{U}$, which is expected to be provided together with the pair (V, sig) in the CA's response. In this scenario, if $h = \mathcal{H}(cert)$, the attacker could replace the CA's legitimate response $(V, sig, \mathcal{U})$ with $(V, sig', \mathcal{U}')$, where $sig' = sig + h\cdot z$ and $\mathcal{U}' = \mathcal{U} + z\cdot G$ for an arbitrary z. The user would be able to reconstruct its implicit keys and verify the correctness of the provided credential, since we would then have $\mathrm{U} = \mathrm{u}\cdot G = (x + sig')\cdot G = (x + r + h\cdot u + h\cdot z)\cdot G = (x\cdot G + r\cdot G) + h\cdot(u\cdot G) + z\cdot G) = V + h\cdot(\mathcal{U} + z\cdot G) = V + h\cdot\mathcal{U}'$.

Hence, the attacker is able to convince *usr* to use the wrong CA public key even without knowing the corresponding private key u' (or, equivalently, the legitimate $u = u' - z$). Such inconsistency issue may not be critical in some contexts, in particular when users know the CA's public key beforehand. Nevertheless, ensuring the consistency of the scheme by design is at least advisable. After all, this avoids subtle attacks that might be build against different systems adopting the proposed implicit certification scheme.

An efficient and secure approach to address this matter consists in including the signer's public key as part of the hash computation, as done in the ISO-9798-3 standard [17, 18] and discussed in [19, 20]. The processing cost of this "key prefixing" technique is expected to be very low, since it corresponds simply to a small increase in the amount of data fed to the underlying hash function. The security benefits, on the other hand, are easily noticeable: as long as $h = \mathcal{H}(cert,\mathcal{U}) \neq \mathcal{H}(cert,\mathcal{U}')$, any attempt to replace $\mathcal{U}$ with $\mathcal{U}'$ in the CA's response would invalidate the CA's signature over h. In addition, as shown in Sections 4.3 and 5.2, enforcing this technique when signing data (not only certificates) is of particular importance to avoid forgery attacks.

4.1 Performance considerations

In the resulting Schnorr-based implicit certification (SIMPL, for short) approach, the implicit certificate's hash h is multiplied by the CA's long-term private key u, instead of the ephemeral private key r. As a result, the public key reconstruction algorithm $\mathrm{U} \overset{?}{=} V + h \cdot \mathcal{U}$ requires one EC multiplication by what is commonly a fixed point: the CA's public key $\mathcal{U}$. This operation can, thus, take full advantage of optimizations such as pre-computation or dedicated code.

For instance, suppose that h is 256 bits long and written in hex, namely $h = h_{63} \cdot 16^{63} + h_{62} \cdot 16^{62} + ... + h_1 \cdot 16 + h_0$ with $0 \leqslant h_p < 16$ for all $0 \leqslant p < 64$. Then, by precomputing and storing a single, fixed 64×16 table $T[p][d] := (d \cdot 16^p) \cdot \mathcal{U}$, the $h \cdot \mathcal{U}$ operation could be implemented via table lookups as follows:

$$\begin{aligned} h{\cdot}\mathcal{U} &= (h_{63}{\cdot}16^{63} + h_{62}{\cdot}16^{62} + ... + h_1{\cdot}16 + h_0){\cdot}\mathcal{U} \\ &= (h_{63}{\cdot}16^{63}){\cdot}\mathcal{U} + (h_{62}{\cdot}16^{62}){\cdot}\mathcal{U} + ... + (h_1{\cdot}16){\cdot}\mathcal{U} + h_0{\cdot}\mathcal{U} \\ &= T[63][h_{63}] + T[62][h_{62}] + ... + T[1][h_1] + T[0][h_0] \end{aligned}$$

Ignoring the (usually small) cost for the table look-ups themselves, the total, fixed cost of this approach would be only 63 point additions. In comparison, a regular double-and-add method would take 255 point doublings, plus an average of 128 additions (or roughly 85 additions if a Non-Adjacent Form (NAF) representation is adopted). Thus, this table-based technique is expected to be about 3 to 4 times faster at the cost of storing a fixed 32 KiB table, considering a classical 128-bit security level with 256-bit elliptic curve finite fields. Other techniques, such as the fixed-base comb method [12], can offer further storage-processing trade-offs. For example, using the fixed comb method with $w = 8$ as implemented in RELIC cryptography library v0.4.1 [21], we were able to run fixed-point multiplications ≈ 8 times faster than random-point multiplications on an Intel i5 4570 processor. It is also worth noting that a side-channel resistant implementation [22] is not necessary for this operation, since all values involved (h and $\mathcal{U}$) are public.

Besides this performance gain, the proposed approach also slightly improves the efficiency of two other processes. The first is the private key reconstruction by the user: since this key is computed simply as $\mathrm{u} \leftarrow x + sig$, instead of $\mathrm{u} \leftarrow h{\cdot}x + sig$ as in ECQV, it saves one modular multiplication. The second is the CA's signing procedure itself, computed as $sig \leftarrow r + h{\cdot}u$ rather than as $sig \leftarrow h{\cdot}r + u$: even though both equations involve the same number of operations, our proposal can benefit from precomputation techniques similar to those applicable to public key reconstruction algorithm. After all, the modular multiplication is such that one of its factors (namely, the private key u) is usually one and the same for all users. In this case, however, the optimization approach must be side-channel resistant, because the fixed factor is private.

4.2 Security Analysis

Assuming the hardness of the elliptic curve discrete logarithm problem (ECDLP) [23], the security properties of the SIMPL scheme should be similar to those provided by ECQV. This claim can be proven under the general security model for implicit certificates described in [3]. This model, illustrated in Figure 1, can be formulated according to the following Definition 1.

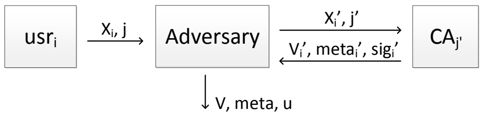

Figure 1: Security model for implicit certificates. Adapted from [3].

Definition 1. (Implicit certificate adversaries [3]) Assume a scenario with n_{usr} legitimate users, denoted usr_i for $1 \leqslant i \leqslant n_{usr}$, and with n_{CA} CAs, denoted CA_j for $1 \leqslant j \leqslant n_{\text{CA}}$. Let (X_i, j) denote usr_i's implicit certificate request for CA_j, and let $(V_i, \texttt{meta}_i, sig_i)$ be the response sent by that CA. Also, let U_i and u_i denote, respectively, the public and the private keys reconstructed by usr_i from CA_j's response, using that CA's public key $\mathcal{U}_j$. There are no restrictions on the number of certificate requests that can be sent by usr_i to CA_j.

A (τ, ϵ)-adversary $\mathcal{A}$ of an implicit certificate scheme is a probabilistic Turing machine that runs in time at most τ and interacts with legitimate users and CAs by performing each of the following operations, any number of times.

(I) receive a request (X_i, j) from usr_i, intended for CA_j; and

(II) send a request $(X'_{i'}, j')$ to $\text{CA}_{j'}$, receiving $(V'_{i'}, \texttt{meta}'_{i'}, sig'_{i'})$ as that CA's response.

With probability at least ϵ, $\mathcal{A}$ outputs a triple $(V, \texttt{meta}, \text{u})$ such that u is the private key corresponding to the public key U reconstructed from V, meta and some $\mathcal{U}_z$ — in our case, this means that $\text{u}{\cdot}G = V + \mathcal{H}(V, \texttt{meta}, \mathcal{U}_z) \cdot \mathcal{U}_z$, — and either

(I) [Forgery attack against CA_z]: $(V, \texttt{meta})$ was never part of a response by CA_z; or

(II) [Key compromise against usr_i]: $(V, \texttt{meta})$ was included in a response of CA_z to some request (X_i, j) originally made by usr_i.

A (τ, ϵ)-adversary $\mathcal{A}$ is considered successful if ϵ is non-negligible for a polynomial time τ.

In summary, this model covers a scenario in which the attacker $\mathcal{A}$ acts as intermediate for the requests from users and responses from CAs. Hence, $\mathcal{A}$ can: simply relay the request to the correct CA; modify the point X_i in the request; modify the user identifier i in the request, thus affecting the value of meta in the certificate; and/or forward the request to a different CA.

Theorem 1 summarizes the security of the SIMPL protocol in this security model.

Theorem 1. *In the random oracle model and assuming the intractability of the ECDLP problem in $\mathbb{G}$, there is no adversary $\mathcal{A}$ that is successful against the hereby proposed Schnorr-based implicit certification (SIMPL) scheme in the sense of Definition 1.*

Proof. The proof is a simple adaptation of [3, Theorem 2] to a Schnorr-style signature scheme, which is known to be secure in the random oracle model assuming the intractability of the ECDLP [24]. Since this adaptation is quite straightforward, we leave the details to the Appendix. □

4.3 Composability with signature schemes: key prefixing

Like other implicit schemes (e.g., ECQV), SIMPL is not claimed to be universally composable. Therefore, using the resulting implicit keys in combination with other cryptographic primitives may require an additional security analysis.

In the specific context of V2X communications, though, implicitly certified keys need basically to be used for signatures. In particular, SCMS recommends the ECDSA [25] signature algorithm, although Schnorr variants may also be of interest. Fortunately, as shown in [14], Schnorr-based implicit certificates can be combined with ECDSA or Schnorr signatures in a secure fashion as long as the key-prefixing technique is employed when signing messages. More precisely, following the recommendation discussed in Section 4, the implicitly-certified key U can be used with such signature schemes if, instead of signing a message M directly, what is signed is the combination (M, U). Alternatively, the signature could be computed over (M, h), where $h = \mathcal{H}(V, \texttt{meta}, \mathcal{U})$, as also proposed in [14]. Both alternatives have an equivalent effect, of binding the signature to a particular public key, thus preventing attacks such as those described in [11]. However, the latter approach is preferable for implicit certificates, since it potential delivers better performance. The reason is that, while h always needs to be computed anyway by the signature verifier, this is not always the case for $\mathrm{U} = V + h{\cdot}\mathcal{U}$. For example, suppose that $\alpha{\cdot}\mathrm{U}$ needs to be computed as part of the signature verification. Then it may be faster to compute $\alpha{\cdot}V + \alpha{\cdot}h{\cdot}\mathcal{U}$ directly, rather than first retrieving U and only then performing the scalar multiplication.

5 Application to Vehicular Communications

Vehicle-to-everything (V2X) communications refers to the set of technologies that allow vehicles to exchange messages among themselves and with other entities (e.g., roadside units or pedestrians) [26]. Its wide scale deployment is motivated by several factors, including improving transportation safety and efficiency. For example, if vehicles exchange Basic Safety Messages (BSM) [27], informing their relative speed, acceleration and distance to each other, accidents can be avoided with or without the drivers' direct intervention. In addition, V2X technologies enable applications in which vehicles and drivers can be informed about adverse road conditions, such as nearby accidents, congestion, busy intersections, slippery conditions, or potholes. Such capabilities create a much more dynamic and information-rich environment than what could be achieved with traditional traffic signs.

An important challenge in V2X environments, though, is to build a Vehicular Public Key Infrastructure (VPKI) able to ensure that authorized vehicles cannot be tracked, either by eavesdroppers or by the system itself [28]. One common approach for this issue is to provision vehicles with multiple, short-lived pseudonym certificates [29]. Vehicles can then avoid tracking by frequently changing the certificates employed to sign their messages while they move. The reasoning is that messages from the same and from different vehicles should not be distinguishable from each other, since none of them carry any identity information. Hence, messages broadcast from different locations and using distinct certificates cannot be easily linked to any given user. On the other hand, the total number of certificates valid simultaneously should be limited (e.g., to a number between 20 and 40 [7]) to avoid misbehavior. One example is a Sybil attack [30], in which one vehicle pretends to be multiple entities. This may allow the culprit, for example, to get preferential treatment from congestion-aware traffic lights [31].

Among the existing VPKI solutions, the Security Credential Management System (SCMS) [7] is of especial interest. Developed in cooperation with the United States Department of Transportation, SCMS's certificate issuance approach combines privacy and scalability in the so-called butterfly key expansion process. Essentially, this process enables several pseudonym certificates to be provisioned with a single request from a vehicle, in such a manner that those certificate cannot be linked to the requester. The resulting architecture is quite bandwidth- and processing-efficient even when issuing thousands of certificates to a vehicle. This is particularly true if the unified butterfly key (UBK) optimization [8] is employed, since it reduces the amount of data exchanged and also the number of operations performed during the issuance process.

In this section, we show that SCMS can benefit from the hereby proposed implicit certification approach. Specifically, we present a novel existential forgery attack that can be mounted against butterfly keys under certain conditions; the attack builds upon the certificate misbinding issue described in Section 4, and can be addressed similarly. In addition, we evaluate the performance gains resulting from integrating SIMPL with the UBK provisioning scheme.

5.1 The unified butterfly key (UBK) expansion process

The certificate provisioning process in SCMS involves mainly three entities:

- Vehicle: the entity that requests pseudonym certificates from a registration authority (RA). For better efficiency, each request leads to the provisioning of a batch containing β certificates.
- Registration Authority (RA): acts as a proxy between vehicles and the Pseudonym Certificate Authority (PCA). Each request from an authorized vehicle generates β individual pseudonym certificate requests sent to the RA, each one containing a different public key. To improve the vehicles' privacy, requests associated with different vehicles are shuffled together by the RA. This prevents the PCA from identifying which requests belong to the same batch, assuming there is no PCA-RA collusion.
- Pseudonym Certificate Authority (PCA): responsible for issuing pseudonym certificates upon request by the RA. The public key effectively placed into the certificate is a randomized version of the key received from the RA, and the response is also encrypted so only the vehicle can retrieve its contents. This prevents the RA from identifying the owner of a certificate when it is used on the field, once again assuming the RA does not collude with the PCA.

Table 4 details the interaction among these entities during the UBK certificate provisioning process. All communications are made via a secure channel, using standard protocols such as Transport Layer Security (TLS). The top rows in this table shows the explicit and implicit models described in [8]. The bottom row shows the hereby proposed implicit approach, highlighting the main differences from regular implicit certificates in gray background.

In all cases, the vehicle starts by picking a random *caterpillar* private key x and computing the corresponding *caterpillar* public key $X = x \cdot G$. The vehicle then sends X, together with a pseudorandom instance f, to the RA.

In response to the vehicle's request, the RA expands the caterpillar public key X into β *cocoon* public keys $\hat{X}_i = X + f(i) \cdot G$. Since f is shared only among the vehicle and the RA, the resulting cocoon keys are unlinkable to the original X from the perspective of any entity other than the vehicle and the RA. The RA then sends each individual $\hat{X}_i$ to the PCA, while shuffling together requests associated to different batches to ensure their unlinkability.

The PCA, in turn, randomizes $\hat{X}_i$ by adding $r_i \cdot G$ to it, for a randomly picked r_i. For explicit certificates, the resulting elliptic curve point is used directly as the vehicle's *butterfly* public key U_i; it is then placed into a certificate together with any required metadata (e.g., a validity period), and signed. For implicit certificates, the randomized point is used as the butterfly reconstruction credential V_i; it is also combined with some metadata, and then signed according to the procedure described in Sections 3 and 4 (depending on the implicit model adopted). In all cases, the resulting certificate is encrypted with the

Table 4: The unified butterfly key (UBK) certificate provisioning process: the top rows show the explicit and implicit certification models from [8]; the bottom row shows the hereby described SIMPL protocol (the gray background highlights the main differences from ECQV implicit certificates).

	Vehicle	$\rightarrow$	RA	$\rightarrow$	PCA	-RA→	Vehicle
(explicit)					$r_i \xleftarrow{\$} \mathbb{Z}_q$ $\mathrm{U}_i \leftarrow \hat{X}_i + r_i \cdot G$ $sig_i \leftarrow Sign(u, \{\mathrm{U}_i, \mathtt{meta}\})$ $cert_i \leftarrow \{\mathrm{U}_i, \mathtt{meta}, sig_i\}$ $pkg \leftarrow Enc(\hat{X}_i, \{cert_i, r_i\})$		$\hat{x}_i \leftarrow x + f(i)$ $\{cert_i, r_i\} \leftarrow Dec(\hat{x}_i, pkg)$ $Verif(\mathcal{U}, cert_i)$ $\mathrm{u}_i \leftarrow \hat{x}_i + r_i$ $\mathrm{u}_i \cdot G \overset{?}{=} \mathrm{U}_i$
(implicit) ECQV	$x \xleftarrow{\$} \mathbb{Z}_q$ $X \leftarrow x \cdot G$	X, f	$\hat{X}_i \leftarrow X + f(i) \cdot G$ $(0 \leqslant i < \beta)$	$\hat{X}_i$	$r_i \xleftarrow{\$} \mathbb{Z}_q$ $V_i \leftarrow \hat{X}_i + r_i \cdot G$ $cert_i \leftarrow \{V_i, \mathtt{meta}\}$ $h_i \leftarrow \mathcal{H}(cert_i)$ $sig_i \leftarrow h_i \cdot r_i + u$ $pkg \leftarrow Enc(\hat{X}_i, \{cert_i, sig_i\})$	pkg	$\hat{x}_i \leftarrow x + f(i)$ $\{cert_i, sig_i\} \leftarrow Dec(\hat{x}_i, pkg)$ $h_i \leftarrow \mathcal{H}(cert_i)$ $\mathrm{u}_i \leftarrow h_i \cdot \hat{x}_i + sig_i$ $\mathrm{U}_i = \mathrm{u}_i \cdot G \overset{?}{=} h_i \cdot V_i + \mathcal{U}$
(implicit) SIMPL					$r_i \xleftarrow{\$} \mathbb{Z}_q$ $V_i \leftarrow \hat{X}_i + r_i \cdot G$ $cert_i \leftarrow (V_i, \mathtt{meta})$ $h_i \leftarrow \mathcal{H}(cert_i, \mathcal{U})$ $sig_i \leftarrow r_i + h_i \cdot u$ $pkg \leftarrow Enc(\hat{X}_i, \{cert_i, sig_i\})$		$\hat{x}_i \leftarrow x + f(i)$ $\{cert_i, sig_i\} \leftarrow Dec(\hat{x}_i, pkg)$ $h_i \leftarrow \mathcal{H}(cert_i, \mathcal{U})$ $\mathrm{u}_i \leftarrow \hat{x}_i + sig_i$ $\mathrm{U}_i = \mathrm{u}_i \cdot G \overset{?}{=} V_i + h_i \cdot \mathcal{U}$

originally provided $\hat{X}_i$, and sent back to the RA. The RA, unable to decrypt the PCA's response *pkg*, simply forwards it back to the requesting vehicle, in batch.

Finally, the vehicle computes the key $\hat{x}_i = x + f(i)$ for decrypting *pkg*. It then verifies that the retrieved certificate is indeed valid. This is done either by checking its signature (for explicit certificates) or by performing the corresponding key verification process (for implicit certificates). As long as such verification is successful, the keys obtained can be used for signing messages sent to other vehicles.

Table 4 summarizes the previous versions of the explicit and implicit protocols, and describes the proposed variant adopting Schnorr-style signatures. From a security point of view, the three certification approaches are basically equivalent. However, the Schnorr variant has efficiency advantages, as discussed in the following subsections.

5.2 The risk of certificate misbinding: building (and fixing) related-key existential forgery attacks

The security of the original butterfly key expansion is discussed in [7, 32, 33], whereas additional security aspects related to the UBK optimization are detailed in [8]. However, none of these works tackle issues arising from one inherent property of butterfly keys: the correlation among different keys from the same batch. More precisely, any pair of private keys $(\mathrm{u}_i, \mathrm{u}_j)$ created from the same caterpillar key x end up sharing some well-defined relationship $\rho_{i,j} = \mathrm{u}_i - \mathrm{u}_j$. Indeed, for explicit certificates we have $\mathrm{u}_i = x + f(i) + r_i$ and $\mathrm{u}_j = x + f(j) + r_j$, meaning that $\rho_{i,j} = f(i) + r_i - f(j) - r_j$. For the hereby described SIMPL certificates, the relationship is quite similar, given by $\rho_{i,j} = f(i) + sig_i - f(j) - sig_j$. In contrast, for regular implicit certificates, this relationship is more complex: since $\mathrm{u}_i = h_i \cdot (x + f(i)) + sig_i$, we can write $\rho_{i,j} = h_i \cdot (x + f(i)) + sig_i - h_j \cdot (x + f(j)) - sig_j$.

In principle, such relationships are unknown by any entity except the owner of the keys. The reason is that $f(i)$ is a secret shared between vehicle and RA, whereas r_i and sig_i (respectively, for explicit and implicit certificates) are known only by vehicle and PCA. Hence, even though $h_i = \mathcal{H}(cert_i)$ is public, the randomness added by PCA and RA is enough to protect the correlation between any pair of keys, in all certification models.

However, it is possible to exploit such relationship among butterfly keys in at least one particular condition: (1) the PCA colludes with the RA, so together they learn the correlation between those keys; (2) a Schnorr-style signature algorithm is employed; and (3) the signatures are prone to certificate misbinding attacks analogous to the one described in Section 4. Condition 1 applies for the explicit and for the hereby described SIMPL certification approaches, since in both cases $\rho_{i,j}$ does not depend on the value of x (known only by the vehicle). Hence, whenever Conditions 2 and 3 are met, the system becomes vulnerable to what we call a “related-key existential forgery” attack, defined as follows:

Definition 2. *(Related-key Existential Forgery)* Suppose that two private keys u_i and u_j share some mathematical relationship that is known by the adversary, although the actual value of those keys are unknown. Given one or more valid signatures generated with u_i, the adversary is able to forge the signature of one message, not necessarily of his/her choice, as if such forgery was generated with u_j.

To understand the attack, it is useful to recall the process by means of which Schnorr digital signatures are generated using private key u and verified with U. First, pick a secret random number α and compute the elliptic curve point $P = \alpha \cdot G$. To sign message m, compute the hash $\mu = \mathcal{H}(P, m)$, and then make $sig(\mu) = \alpha + \mu \cdot \mathrm{u}$). The output of the signature process is the pair $(\mu, sig(\mu))$. The verification process then consists in recovering the elliptic curve point $P = sig(\mu) \cdot G - \mu \cdot \mathrm{U}$ and checking for the equality $\mathcal{H}(P, \mu) \overset{?}{=} \mu$.

For the butterfly key u_i, the signature $(\mu, sig(\mu)_i)$ is such that $\mu = \mathcal{H}(P, m)$ and $sig(\mu)_i = \alpha + \mu \cdot \mathrm{u}_i$). It turns out that, given this signature, the colluding RA and PCA can easily compute the signature for $(\mu, sig(\mu)_j)$, i.e., the signature for message m as if it was produced with u_j. For this, they have to first recover $P = sig(\mu)_i \cdot G - \mu \cdot \mathrm{U}_i$ and then simply compute $sig(\mu)_j = sig(\mu)_i - \mu \cdot \rho_{i,j}$. This is a valid signature because $sig(\mu)_j = \alpha + \mu \cdot \mathrm{u}_j = (\alpha + \mu \cdot \mathrm{u}_i) - \mu \cdot \mathrm{u}_i + \mu \cdot \mathrm{u}_j = sig(\mu)_i - \mu \cdot \rho_{i,j}$. Both signatures refer to the same vehicle, also identifiable due to the collusion, so this attack allows RA and PCA to frame a target for sending m using different identities. As a possible consequence, vehicles could be wrongly accused of performing Sybil-like attacks [30]. Hence, the risks of an RA-PCA collusion would go beyond the system's privacy (i.e., the ability to track vehicles), but also affect its security.

Fortunately, since this issue is basically an extension of a certificate misbinding attack, it can be averted using the same technique recommended in Section 4.3: via key prefixing. Indeed, with this approach the signature generated with u_i would be $(\mu_i, sig(\mu)_i)$, where $\mu_i = \mathcal{H}(P, \mu, h_i)$, $h_i = \mathcal{H}(cert_i)$, and $sig(\mu)_i = \alpha + \mu_i \cdot \mathrm{u}_i$). To employ the same trick as before, the attacker would then have to compute $sig(\mu)_j = sig(\mu)_i - \mu_i \cdot \mathrm{u}_i + \mu_j \cdot \mathrm{u}_j = sig(\mu)_i - \mu_i \cdot \rho_{i,j} + (\mu_j - \mu_i) \cdot \mathrm{u}_j$, which should be unfeasible without knowing u_j (or, equivalently, u_i).

At this point, it is worth noting that such attack does not invalidate SCMS's security claims, for at least three reasons. The first is that one of SCMS's security assumptions is that PCA and RA do not collude, nor engage in active attacks. In addition, SCMS recommends using ECDSA [25] as underlying signature algorithm, for which we were unable to reproduce the hereby presented attack. Finally, the latest version of SCMS already suggests the countermeasure hereby proposed, i.e., including the signer's certificate information in the hash computation, although this recommendation was motivated by what was considered a "not particularly significant attack" [33, Section V.D]. All in all, and to ensure that SCMS remains secure against forgery even if vehicles use a Schnorr variant for signing messages, it is sensible to mandate that all signature made with butterfly keys include this countermeasure.

5.3 Experimental results

To evaluate the performance gains provided by the proposed the SIMPL certification approach, we use an experimental setup similar to the one described in the UBK paper

Table 5: Processing costs (in cycles) for the UBK approach when issuing when β certificates with the explicit, regular implicit and proposed implicit approaches.

	Vehicle	$\rightarrow$	RA	$\rightarrow$	PCA	-RA→	Vehicle
UBK explicit	254×10^3	-	$\beta \cdot (250 \times 10^3)$	-	$\beta \cdot (2.86 \times 10^6)$	-	$\beta \cdot (2.73 \times 10^6)$
UBK implicit (original)					$\beta \cdot (2.47 \times 10^6)$		$\beta \cdot (4.19 \times 10^6)$
UBK implicit (proposed)					$\beta \cdot (2.47 \times 10^6)$		$\beta \cdot (2.45 \times 10^6)$
Proposed/explicit	1		1		1		0.91
Proposed/original							0.58

[8]. Namely, we employ the RELIC cryptography library version 0.4.1 [21] on an Intel i5 4570 processor for implementing the explicit, regular implicit and proposed implicit UBK protocol. We then measure the number of cycles on all entities involved in the certificate provisioning process. We do so mainly for completeness, though, since the only non-negligible performance gains of the proposed approach is on the vehicles' side, when they verifies their own or their peer's public key reconstruction credentials. The underlying algorithms adopted are those recommended by SCMS's proof-of-concept implementation [32]: ECDSA [25] for signature generation/verification and ECIES [34] for asymmetric encryption/decryption, both configured with a 128-bit security level.

In all implementations, we assume that the PCA's public key $\mathcal{U}$ can be considered a fixed point and, thus, can be optimized as discussed in Section 4.1. We argue that this is a reasonable assumption in real-world V2X deployments because, even if different PCAs are employed for better privacy [8], the total number of PCAs is expected to remain small in practice. Therefore, the vehicles' memory capabilities are likely to support the resulting (few) pre-computation tables required by such optimization.

The results obtained for the average of 1000 executions of each operation, which leads to a standard deviation below 1%, are depicted in Table 5. As shown in this table, the processing cost of the proposed scheme is reasonably lower at the vehicle. namely, the costs of the proposed implicit approach is 91% and 58% of the cost taken, respectively, by UBK in the explicit and in the original implicit model.

Finally, it is interesting to notice that the adoption of implicit certificates in SCMS seems to have been motivated for the two reasons highlighted in [35, page 17]: (1) it potentially leads to smaller certificates and, hence, to lower bandwidth usage in V2X communications; and (2) during operation, the process of verifying a message's signature can be combined with public key verification for better computational efficiency. Nevertheless, when compared to explicit certificates, the original implicit model leads to a more costly batch verification procedure: as shown in Table 5, the processing costs at the vehicle's side with the original implicit certificates is roughly 1.53$\times$ the one obtained in the explicit model. This potential source of criticism is overcame by the proposed SIMPL scheme, though, since in this case the implicit approach is actually less costly than the explicit counterpart. Therefore, our proposal allows SCMS to fully benefit from the implicit certification model's bandwidth savings.

6 Conclusion

The main benefit of the implicit certification model is that it leads to smaller certificate material than traditional approaches. In many cases, however, this comes at the cost of increased processing when verifying the validity of implicitly certified public keys.

In this article, we show discuss an implicit certification scheme that tackles the main bottleneck of implicit certification schemes such as [4]: the costly scalar multiplication by elliptic curve (EC) points involved in the public key verification process. Specifically, by using an OMC-like, Schnorr-based signature approach for certifying keys, random-point EC multiplications can be replaced by fixed point operations. The resulting scheme supports, thus, optimized implementations that in principle cannot be applied to traditional implicit certificates.

We also evaluate the actual gains of the proposed implicit certification approach in the context of vehicular communications. Namely, we benchmark an implementation of SCMS [7] with the UBK optimization [8] considering the explicit model, the ECQV scheme, and the proposed SIMPL implicit certification approach. Our benchmark results show that the scheme hereby described leads to better performance than the other alternatives, while preserving the bandwidth savings typical of implicit certificates.

As an additional contribution, we present a novel existential forgery attack that can be mounted against butterfly keys under certain conditions. The attack exploits the underlying properties of butterfly keys, namely, the correlation between pairs of keys issued to the same vehicle in a given batch. Therefore, this vulnerability applies not only to the original SCMS described in [7], but it is also inherited by the UBK optimization given in [8]. We also show a simple fix for this issue, which was already suggested in the most recent SCMS specification given in [33], although for different reasons.

Acknowledgment. This work was supported by LG Electronics, via the Foundation for the Technological Development of the Engineering Sciences (FDTE). It was also funded in part by the Brazilian CAPES (Finance Code 001), CNPq (grants 301198/2017-9 and 132485/2017-6) and FAPESP (grant 13/25977-7). We would also like to thank Zhaohui Cheng and Paul Bottinelli for their useful comments on an early version of this manuscript.

References

[1] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) profile," https://tools.ietf.org/html/rfc5280#section-4.2.1.3, May 2008.

[2] J. K. Liu, M. H. Au, and W. Susilo, "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model: Extended abstract," in *Proc. of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS'07).* New York, NY, USA: ACM, 2007, pp. 273–283.

[3] D. Brown, R. Gallant, and S. Vanstone, "Provably secure implicit certificate schemes," in *Proc. of the 5th International Conference on Financial Cryptography (FC'01).* Berlin, Heidelberg: Springer-Verlag, 2002, pp. 156–165.

[4] Certicom, "SEC 4: Elliptic curve Qu-Vanstone implicit certificate scheme (ECQV)," Certicom Research, Tech. Rep., 2013, www.secg.org/sec4-1.0.pdf.

[5] S. Sciancalepore, A. Capossele, G. Piro, G. Boggia, and G. Bianchi, "Key management protocol with implicit certificates for IoT systems," in *Proc. of the 2015 Workshop on IoT Challenges in Mobile and Industrial Systems (IoT-Sys'15).* New York, NY, USA: ACM, 2015, pp. 37–42.

[6] M. Simplicio, M. Silva, R. Alves, and T. Shibata, "Lightweight and escrow-less authenticated key agreement for the Internet of Things," *Computer Communications*, vol. 98, pp. 43–51, 2017.

[7] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn, "A security credential management system for V2V communications," in *IEEE Vehicular Networking Conference (VNC'13)*, 2013, pp. 1–8.

[8] M. Simplicio, E. Cominetti, H. Kupwade-Patil, J. Ricardini, and M. Silva, "The unified butterfly effect: Efficient security credential management system for vehicular communications," in *IEEE Vehicular Networking Conference (VNC'18)*, 2018, see also: eprint.iacr.org/2018/089.pdf.

[9] T. Weil, "VPKI hits the highway — secure communication for the US DOT connected vehicle pilot program," https://www.securityfeeds.us/sites/default/files/VPKI_Hits_Highway_DineLearn_9May2017.pdf, May 2017, iEEE COMSOC Presentation.

[10] L. Pintsov and S. Vanstone, "Postal revenue collection in the digital age," in *Financial Cryptography*. Berlin, Heidelberg: Springer, 2001, pp. 105–120.

[11] D. Brown, M. Campagna, and S. Vanstone, "Security of ECQV-certified ECDSA against passive adversaries," Cryptology ePrint Archive, Report 2009/620, 2009, https://eprint.iacr.org/2009/620.

[12] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[13] C. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[14] Z. Cheng and L. Chen, "Certificateless public key signature schemes from standard algorithms," in *Information Security Practice and Experience*. Cham: Springer International Publishing, 2018, pp. 179–197.

[15] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.

[16] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols," in *Advances in Cryptology - CRYPTO 2003*. Berlin, Heidelberg: Springer, 2003, pp. 400–425.

[17] ISO, *ISO/IEC 9798-3 – IT Security techniques – Entity authentication – Part 3: Mechanisms using digital signature techniques*, International Organization for Standardization, 1993. [Online]. Available: https://www.iso.org/standard/67115.html

[18] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Advances in Cryptology — EUROCRYPT 2001*. Berlin, Heidelberg: Springer, 2001, pp. 453–474.

[19] A. Menezes and N. Smart, "Security of signature schemes in a multi-user setting," *Designs, Codes and Cryptography*, vol. 33, no. 3, pp. 261–274, Nov 2004.

[20] D. Bernstein, "Multi-user schnorr security, revisited," Cryptology ePrint Archive, Report 2015/996, 2015, https://eprint.iacr.org/2015/996.

[21] D. Aranha and C. Gouvêa, "RELIC is an Efficient LIbrary for Cryptography," https://github.com/relic-toolkit/relic, 2018.

[22] B. Möller, "Securing elliptic curve point multiplication against side-channel attacks," in *Information Security (ISC'01)*. Berlin, Heidelberg: Springer, 2001, pp. 324–334.

[23] K. E. Lauter and K. E. Stange, "The elliptic curve discrete logarithm problem and equivalent hard problems for elliptic divisibility sequences," in *Selected Areas in Cryptography (SAC'08)*. Springer, 2008, pp. 309–327.

[24] Y. Seurin, “On the exact security of Schnorr-type signatures in the random oracle model,” in *Proc. of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT’12).* Berlin, Heidelberg: Springer-Verlag, 2012, pp. 554–571.

[25] NIST, *FIPS 186-4: Digital Signature Standard (DSS)*, National Institute of Standards and Technology, July 2013. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[26] ETSI, “Intelligent transport systems (ITS), vehicular communications, basic set of applications – part 2: Specification of cooperative awareness basic service,” https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.01_30/en_30263702v010301v.pdf, European Telecommunications Standards Institute, Tech. Rep., Sep 2014.

[27] S. Banani and S. Gordon, “Selecting basic safety messages to verify in vanets using zone priority,” *The 20th Asia-Pacific Conference on Communication (APCC2014)*, pp. 423–428, 2014.

[28] M. Khodaei and P. Papadimitratos, “The key to intelligent transportation: Identity and credential management in vehicular communication systems,” *IEEE Veh. Technol. Mag.*, vol. 10, no. 4, pp. 63–69, Dec 2015.

[29] J. Petit, F. Schaub, M. Feiri, and F. Kargl, “Pseudonym schemes in vehicular networks: A survey,” *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 228–255, 2015.

[30] J. Douceur, “The Sybil attack,” in *Proc. of 1st International Workshop on Peer-to-Peer Systems (IPTPS).* Springer, January 2002.

[31] A. K. Sharma, S. K. Saroj, S. K. Chauhan, and S. K. Saini, “Sybil attack prevention and detection in vehicular ad hoc network,” in *International Conference on Computing, Communication and Automation (ICCCA’16)*, 2016, pp. 594–599.

[32] CAMP, “Security credential management system proof–of–concept implementation – EE requirements and specifications supporting SCMS software release 1.1,” Vehicle Safety Communications Consortium, Tech. Rep., may 2016.

[33] B. Brecht, D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, and R. Goudy, “A security credential management system for V2X communications,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3850–3871, Dec 2018.

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

[35] Certicom, “Sec 4 v1.0: Elliptic curve Qu-Vanstone implicit certificate scheme (ECQV),” Certicom Research, Tech. Rep., 2013, http://www.secg.org/sec4-1.0.pdf.

A Appendix

In this appendix, we provide the proof for Theorem 1, showing the security of the proposed Schnorr-based implicit certification scheme. The proof is largely based on the security model proposed on [3], which is graphically illustrated in Figure 1. We note that this is also the security model commonly used by regular implicit certificates [4].

We hereby assume the following process for generating Schnorr digital signatures under private key u and verifying them with the corresponding public key $\mathrm{U} = \mathrm{u} \cdot G$. The signer starts by picking a secret random number α and computing the elliptic curve point

$P = \alpha{\cdot}G$. To sign message m, the signer computes the hash $\mu = \mathcal{H}(P, m, \mathrm{U})$, and then makes $sig(\mu) = \alpha + \mu{\cdot}\mathrm{u})$. The output of the signature process is the pair (μ, sig), which can be verified by computing $P = sig{\cdot}G - \mu{\cdot}\mathrm{U}$ and then checking whether $\mathcal{H}(P, m, \mathrm{U}) \overset{?}{=} \mu$ holds true. However, this signature form is not the only possibility. A security-equivalent and interchangeable form is the pair (P, sig), whose signature verification consists in computing $\mu \leftarrow \mathcal{H}(P, m, \mathrm{U})$ and then checking whether $sig \cdot G \overset{?}{=} P + \mu \cdot \mathrm{U}$ holds true. For better compatibility with [3, Theorem 2], we adopt this second signature form along the discussion.

Theorem 1. *In the random oracle model and assuming the intractability of the ECDLP problem in $\mathbb{G}$ of prime order q, there is no adversary $\mathcal{A}$ that is successful against the hereby proposed Schnorr-based implicit certification scheme in the sense of Definition 1.*

Proof. We build a proof by contradiction, which is a simple adaptation of [3, Theorem 2] to the proposed Schnorr-style signature scheme. The proof relies on the intractability of the ECDLP [23]. Indirectly, it also relies on the fact that Schnorr signatures are secure against forgery in the random oracle model, also assuming the ECDLP hardness [24].

First, assume that the proposed Schnorr-based implicit certification scheme is not secure although the employed hash function $\mathcal{H}$ can be considered a random oracle. Then, there exists a successful (τ, ϵ)-adversary $\mathcal{A}$ as per Definition 1. More precisely, we can build a polynomial-time algorithm $\mathcal{S}$ that, using $\mathcal{A}$ as subroutine, is able to solve the ECDLP in $\mathbb{G}$ with non-negligible probability.

The input to algorithm $\mathcal{S}$ is a ECDLP challenge $C \in \mathbb{G}$, $C \neq \mathcal{O}$, where $\mathcal{O}$ is the point at infinity. The expected output of $\mathcal{S}$ is an integer $c \in [1, q-1]$ satisfying the discrete logarithm equation, $C = c{\cdot}G$. We build $\mathcal{S}$ in two stages. The first stage, denoted $\mathcal{S}_1$, takes as input $(C, m, \mathcal{H}_1)$ where m is a random message and $\mathcal{H}_1$ is a random oracle independent of $\mathcal{H}$. The desired output of $\mathcal{S}_1$ is either: (i) an integer $c \in [1, q-1]$ satisfying $C = c{\cdot}G$; or (ii) an ordered pair (V, u) such that $\mathrm{u}{\cdot}G = V + \mathcal{H}_1(V, m, C){\cdot}C$, which means that (V, u) is a valid Schnorr signature of message m under the public key C. If case (i) occurs, then $\mathcal{S}$ simply outputs c and terminates. If case (ii) occurs, then $\mathcal{S}_1$'s ability to create a Schnorr signature forgery can be converted into the ability to solve the ECDLP for extracting c [24]. Whichever the case, success at this stage means that $\mathcal{S}$ outputs c and terminates.

To accomplish the task of finding c, $\mathcal{S}_1$ can use $\mathcal{A}$ as a subroutine. Algorithm $\mathcal{A}$ expects the existence of one or more CAs, each with a public key for which $\mathcal{A}$ is not given the private key. It also assumes there are one or more users usr_i making one or more requests containing a public key $X_i = x_i{\cdot}G$ for which $\mathcal{A}$ is not given the corresponding discrete logarithm x_i. Algorithm $\mathcal{S}_1$ randomly picks one CA public keys or one user request to be the challenge point C, which is used as the input for $\mathcal{S}$. Other requested curve points and CA public keys can be chosen by $\mathcal{S}_1$ at will, simply by selecting a random secret integer z and computing $z{\cdot}G$. Let t be the total number of CA public keys and user requests at the attacker's disposal. In what follows, we show that $\mathcal{A}$ can be successfully used by $\mathcal{S}$ to obtain c or a Schnorr signature forgery under public key C with a probability of ϵ/t.

We start by noticing that, if $\mathcal{S}_1$ selects a CA whose public key is C, $\mathcal{A}$ can request a certificate from that CA and receive a legitimate response. Therefore, $\mathcal{S}_1$ must be able emulate this behavior, providing a response that seems legitimate at least from $\mathcal{A}$'s perspective; otherwise $\mathcal{A}$ is not guaranteed success and, thus, it may not be useful for $\mathcal{S}_1$ when trying to find the correct c. However, $\mathcal{S}_1$ has no knowledge of the private key c associated with C. Nevertheless, $\mathcal{S}_1$ can accomplish this by leveraging the fact that the hash function $\mathcal{H}$ is a random oracle. In other words, $\mathcal{S}_1$ can simulate the CA, answering $\mathcal{A}$'s certificate requests with valid responses, by careful pre-selecting the random values output by $\mathcal{H}$. Specifically, algorithm $\mathcal{S}_1$ simulates the CA as follows: given a user request containing the public key X, which is expected to be paired with certificate metadata $\mathtt{meta}_i$, $\mathcal{S}_1$ generates two random integers $sig_i, h_i \in [0, q-1]$ and computes $V_i = X_i + sig_i{\cdot}G - h_i{\cdot}C)$.

Then, $\mathcal{S}_1$ sets $\mathcal{H}(V_i, \texttt{meta}_i, C) = h_i$ and returns the triple $(V_i, \texttt{meta}_i, sig_i)$ as the response to $\mathcal{A}$'s request. This response appears legitimate from $\mathcal{A}$'s perspective, since the public key computed as $\mathrm{U}_i = (x_i + sig_i)\cdot G$ successfully passes the following authenticity check:

$$\begin{aligned}
\mathrm{U}_i &\overset{?}{=} V_i + h_i\cdot C \\
&\overset{?}{=} X_i + sig_i\cdot G - h_i\cdot C + h_i\cdot C \\
&\overset{?}{=} X_i + sig_i\cdot G \\
&\overset{?}{=} (x_i + sig_i)\cdot G \qquad \triangleright \text{ always true, by definition of } \mathrm{U}_i
\end{aligned}$$

Furthermore, the hash function simulates a random oracle from $\mathcal{A}$'s perspective, since the value h_i is chosen at random by $\mathcal{S}_1$.

The adversary $\mathcal{A}$ is, as usual, allowed to query $\mathcal{H}$ directly for any input not previously used in $\mathcal{S}_1$'s responses. For example, suppose that the hash function is queried with input $(V_{\mathcal{A}}, \texttt{meta}_{\mathcal{A}}, C)$, which has not been previously queried or computed by $\mathcal{S}_1$ as above. Then, $\mathcal{S}_1$ outputs $\mathcal{H}_1(V_{\mathcal{A}}, m, C)$ where m is the message whose signature $\mathcal{S}_1$ wishes to forge. In this scenario, the distribution of simulated hash values picked by $\mathcal{S}$ is indistinguishable to $\mathcal{A}$ from the distribution of hash values generated by an actual random oracle.

Now, suppose that algorithm $\mathcal{A}$ is successful, i.e., that $\mathcal{A}$ is able provide a response $(V, \texttt{meta}, \mathrm{u})$ such that $\mathrm{u}\cdot G = V + \mu\cdot C_k$ for some k, where $\mu = \mathcal{H}(V, \texttt{meta}, C_k)$ and either:

(I) $(V, \texttt{meta})$ is a certificate created by CA_k in response to a request from usr_i; or

(II) $(V, \texttt{meta})$ is a certificate that has not been issued by CA_k.

Suppose that case (I) is true. Then, there is a probability of at least $1/t$ that X_i enclosed in a request from usr_i was the challenge point C given as input to algorithm $\mathcal{S}_1$. Since the private key u belonging to usr_i and discovered by $\mathcal{A}$ is valid, it must satisfy $\mathrm{u} = x + sig$. However, in this case we have $c = x$, and $\mathcal{S}_1$ can obtain sig from CA_k's original response. Hence, $\mathcal{S}_1$ is able to compute $c = (\mathrm{u} - sig)$.

Suppose now that case (II) is true. Then, there is a probability of at least $1/t$ that CA_k's public key C_k was the challenge point C given as input to $\mathcal{S}_1$. It is safe to assume that the triple $(V, \texttt{meta}, C)$ was used as input in a query to $\mathcal{H}$, because otherwise the verification equation $\mathrm{u}\cdot G \overset{?}{=} V + \mathcal{H}(V, \texttt{meta}, C)\cdot C$ holds true only with negligible probability, contradicting the assumption that ϵ is non-negligible. Therefore, we have $\mathcal{H}(V, \texttt{meta}, C) = \mathcal{H}_1(V, m, C)$ by definition of the simulation. Consequently, (V, u) is a valid Schnorr signature for message m.

One potential (but minor) issue with this simulation is that, while $\mathcal{S}_1$ executes with $\mathcal{A}$, the pair $(V, \texttt{meta})$ may appear first in a direct query to $\mathcal{H}$, and later as a certificate constructed during the simulation of a CA. Since $\mathcal{S}_1$ picks the values sig_i and h_i at random while simulating the CA, though, the elliptic curve point V_i is expected to be uniformly distributed. Hence, the occurrence of $V = V_i$ should only happen with negligible probability. In addition, if that happens, $\mathcal{S}_1$ can simply start over. Finally, we can conclude that, if $\mathcal{A}$ runs in polynomial time and succeeds with non-negligible probability, then so will $\mathcal{S}_1$. From the above discussion and assuming the security of Schnorr signatures against forgery, if $\mathcal{A}$ runs in polynomial time and succeeds with non-negligible probability, then so will $\mathcal{S}$. However, by hypothesis, it was assumed that no such $\mathcal{S}$ for solving discrete logarithms in $\mathbb{G}$ exists. Therefore, by contradiction, no adversary $\mathcal{A}$ should exist in the random oracle model unless discrete logarithms in $\mathbb{G}$ can be efficiently solved. $\square$