Secure and Efficient Implicit Certificates: Improving the Performance for Host Identity Protocol in IoT

Zhaokang Lu[1] and Jianzhu Lu[2]

[1] School of Computer Science and Technology, Harbin University of Science and Technology, Harbin, 150080, China 2727716939@qq.com

[2] Department of Computer Science, Jinan University, Guangzhou, 510630, China
tljz@jnu.edu.cn

Abstract. Implicit certificates own the shorter public key validation data. This property makes them appealing in resource-constrained IoT systems where public-key validation is performed very often, which is common in Host Identity Protocol (HIP). However, it is still a critical challenge in IoT how to guarantee the security and efficiency of implicit certificates. This article presents a forgery attack for the Privacy-aware HIP (P-HIP), and then propose a Secure and Efficient Implicit Certificate (SEIC) scheme that can improve the security of the P-HIP and the efficiency of elliptic-curve point multiplications for IoT devices. For a fix-point multiplication, the proposed approach is about 1.5 times faster than the method in SIMPL scheme. Furthermore, we improve the performance of SEIC with the butterfly key expansion process, and then construct an improved P-HIP. Experimental results show that the improved P-HIP can achieve the performance gains.

1 Introduction

Public-key validation is a critical issue for any IoT system that relies on public/private key pairs for digital signature, key exchange and/or asymmetric encryption. Many current IoT devices rely on the Public Key Infrastructure (PKI) to achieve public-key validation[1]. Traditionally, this is accomplished by means of explicit certificates, digital documents that enclose the device's public key and are signed by a trusted Certificate Authority (CA). This relies basically on X.509 certificates [2] for verifying the ownership of public keys. For instance, for the authentication and key exchange of HIP shown in Figure 1, the certificates $cert_r$ and $cert_r$ are carried in the messages R_1 and I_2, respectively. Digital signatures of the parties are applied for this situation. In order to verify the correctness of a message signed by the private key of the sender, a receiver first needs to validate the corresponding public-key via its certificate.

This work focuses on a specific yet important problem: how to attain fast public-key validation in IoT. Allowing one IoT device to achieve public key validation remains a challenging problem [1]. This is because certificate verification involves expensive public key encryption and bandwidth cost for certificate transmission. Certificate verification consists of three main steps: (1) check its validity period; (2) validate the CA's digital signature using CA's public key; (3) verify the certificate revocation status. Step

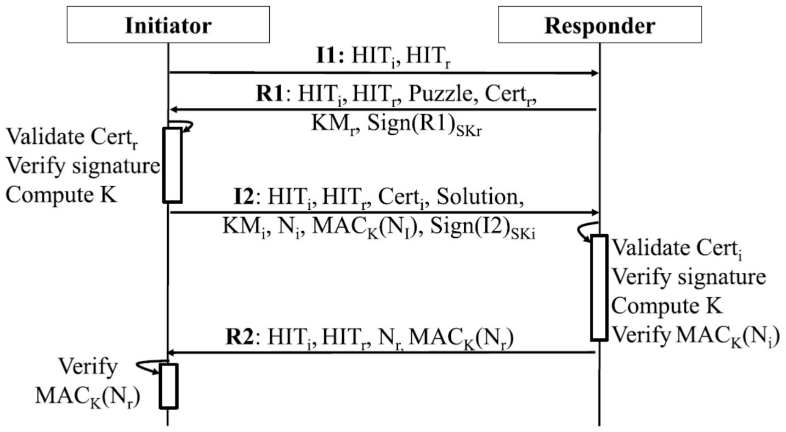

Initiator
Responder
I1: HIT_i, HIT_r
R1: HIT_i, HIT_r, Puzzle, Cert_r, KM_r, Sign(R1)_SKr
Validate Cert_r
Verify signature
Compute K
I2: HIT_i, HIT_r, Cert_i, Solution, KM_i, N_i, MAC_K(N_i), Sign(I2)_SKi
Validate Cert_i
Verify signature
Compute K
Verify MAC_K(N_i)
R2: HIT_i, HIT_r, N_r, MAC_K(N_r)
Verify MAC_K(N_r)

Fig. 1. HIP handshakes for authentication and key exchange.

(1) is simple and Step (3) is discussed in the next paragraph of this section. In particular, step (2) has to perform at least one expensive public key operation. On the other hand, from Table 1 we notice that the certificate size is much larger than the public key size, and their ratio is greater than 5. Some authors [3, 4] explored how to reduce the communication overhead for public-key validation. Compared to traditional explicit certificates, an implicit certificate has the shorter public key validation data. This better efficiency makes implicit certificates appealing in IoT devices. This paper will propose a scheme to further improve the efficiency of public key validation using implicit certificates.

Table 1. X.509 certificate size versus public-key length

RSA				ECC			
KL	KS_1	CS	rat	KL	KS_2	CS	rat
1024	128	1237	9.66	112	28	948	33.85
2048	256	1590	6.21	160	40	981	24,52
3072	384	1935	5.03	256	64	1050	16.40
KL: Key Length(bits); KS_1: Modulus N size(Bytes); CS: Certificate Size(Bytes)							
KS_2: Uncompressed point Q(X,Y) size(Bytes); rat:CS/KS_1 or CS/KS_1							

Fast public-key revocation validation seems to be a dilemma on IoT devices. In PKI, public-key revocation is achieved through the certificate revocation. The unexpired certificates are revoked by using Certificate Revocation Lists (CRLs) or the Online Certificate Status Protocol (OCSP). The CRL introduces substantial communication overhead since the CRL size is proportional to the number of revoked certificates. The OCSP increases certificate revocation verification latency and the risk of leaking user privacy (such as accessing history of the device). Hence none of the above methods is desired for IoT. In order to reduce both the communication overhead and the latency, based on Nyberg's one-way accumulator, this paper designs a credential revocation mechanism which is significantly efficient in the verifier's side.

Privacy protection on IoT devices is a rising issue as identity masquerade and identity tracing have become common attacks in wireless mobile environments. HIP is a suitable solution for IoT devices considering the security and privacy requirements of IoT systems [5]. In the HIP, an IoT device is issued a public key as host identifier and a 128-bits hash of the public key as Host Identity Tag (HIT). A mobile IoT device uses the same public key and HIT to authenticate to its peers when it moves from one network to another network. By learning the public key and HIT, an attacker can track the mobile IoT device. This paper improves the security of P-HIP in [5] such that devices can avoid tracking by changing the short-lived implicit certificates.

The main requirements of practical public-key validation in IoT are summarized as follows: (1)Security: The CA can not be compromised or coerced to assign a public key to a malicious attacker. (2)Accuracy: A device should determine a certificate revocation status without error. (3)User privacy: The protocol should not leak the identities of the accessing devices. (4)Efficiency: The protocol should cost small memory, computation, and network resource on IoT devices. (5)Compatibility: The protocol is required to be compatible with current certificate standards and existing certificates.

Our contribution In this paper, we first present a forgery attack for P-HIP scheme [5], and then propose a secure and efficient implicit certificate (SEIC) scheme to overcome its weaknesses. Specifically, the SEIC scheme runs the signature algorithm to output a signature by hashing the public key validation data, the timestamp and the CA's public key. A table-and-optimality-based technique is designed for Elliptic-Curve (EC) fixed-point multiplication such that it's achievement is about 1.5 times faster than the method [6]. In addition, SEIC contains a credential revocation mechanism which is significantly efficient in the verifier's side. That is, the verifier achieves the revocation verification of a public key by performing one Nyberg one-way accumulator operation while keeping only one $\mathbb{Z}_p$ symbol. Furthermore, we improve SEIC via the butterfly key expansion process, and then construct an improved P-HIP. Experimental results show that the improved P-HIP can achieve performance gains during credential issuance and mutual authentication, while preserving the user privacy.

The rest of this paper is organized as follows: Section 2 analyzes the security of the implicit certificate scheme in P-HIP [5] after an overview of the scheme, and then introduces related work. Section 3 explains the basics notations as well as the primitives of proposed scheme. Section 4 introduces SEIC scheme and discusses its security and performance. Section 5 shows that SEIC can be improved via the butterfly key expansion process, and then an improved P-HIP is constructed. Section 6 formally analyzes the privacy of the improved P-HIP and the corresponding performance gains is shown in Section 7. Finally, Section 8 concludes the discussion.

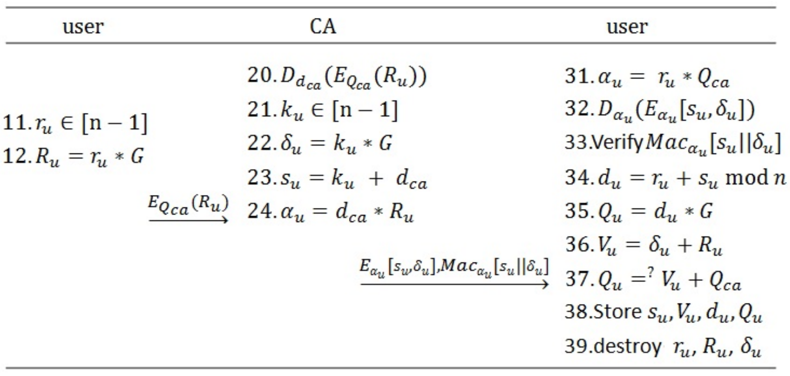
2 Analysis of the implicit certificate scheme in P-HIP and related work

2.1 The implicit certificate scheme in P-HIP

Consider an additive cyclic group $\mathcal{G}$ generated by a point G on the elliptic curve $y^2 = x^3 + ax + b$ over a finite field $\mathbb{F}_q$, where q is a large prime and $4a^3 + 27b^2 \neq 0(\bmod q)$, and n is the order of generator G. We assume that Q_{ca} and $Q_{\overline{ca}}$ in $\mathcal{G}$ are the public keys of CA and $\overline{\text{CA}}$, respectively.

As shown in Figure 2, an implicit certificate scheme was proposed recently to design a P-HIP [5]. We review the scheme as follows. The basic goal of the scheme was to bind a public key Q_u to its owner u via the public-key validation data V_u. To compute the private key construction data s_u, the scheme is different from the conventional ECQV implicit certificate schemes. That is, the CA does not issue a certificate $(cert_u)$ to the user. The scheme computed s_u as $s_u = k_u + d_{ca}(modn)$, which did not multiply h_u=$Hash(cert_u)$ with k_u or d_{ca} to compute s_u (see the steps 23 of Fig. 2). Then, s_u and δ_u were encrypted and then sent to the user. Upon receiving a new ECQV-based credential s_u and δ_u, the user computed a unique public key Q_u=$d_u * G$ and HIT HIT_u=$Hash(Q_u)$ for a network that it would join without communicating with the CA.

A user device provided its ECQV public key Q_u and the public-key validation parameter V_u to a verifier device. The verifier computed a public key Q'_u as Q'_u=$V_u + Q_{ca}$. If Q'_u=Q_u, then the verifier ensured that the public key was genuine and issued by the CA.



user	CA	user
	20.$D_{d_{ca}}(E_{Q_{ca}}(R_u))$	31.$\alpha_u = r_u * Q_{ca}$
11.$r_u \in [\text{n} - 1]$	21.$k_u \in [\text{n} - 1]$	32.$D_{\alpha_u}(E_{\alpha_u}[s_u, \delta_u])$
12.$R_u = r_u * G$	22.$\delta_u = k_u * G$	33.Verify$Mac_{\alpha_u}[s_u \| \delta_u]$
	23.$s_u = k_u + d_{ca}$	34.$d_u = r_u + s_u \bmod n$
$\xrightarrow{E_{Qca}(R_u)}$	24.$\alpha_u = d_{ca} * R_u$	35.$Q_u = d_u * G$
		36.$V_u = \delta_u + R_u$
	$\xrightarrow{E_{\alpha_u}[s_u, \delta_u], Mac_{\alpha_u}[s_u \| \delta_u]}$	37.$Q_u =^? V_u + Q_{ca}$
		38.Store s_u, V_u, d_u, Q_u
		39.destroy r_u, R_u, δ_u

Fig. 2. The implicit certificate scheme in P-HIP [5].

2.2 Security weaknesses of the implicit certificate scheme in P-HIP

We have observed weaknesses of the implicit certificate scheme in P-HIP [5]. First, the scheme suffered from a forgery attack. This is because a malicious user holding the implicit certificate issued by a CA is able to forge an implicit certificate issued by another CA, precluding the use of any digital signature scheme. In the case, as shown in Figure 2, the CA whose public key is Q_{ca} had issued an implicit certificate (V_u, d_u, Q_u) to u. Assume that an adversary either is u himself or colluding with him. Let $\overline{\text{CA}}$ be compromised by the adversary who knows its public key $Q_{\overline{\text{CA}}} = (\overline{x}, \overline{y})$. The adversary disguising as $\overline{\text{CA}}$ can mount a forge attack such that a forged $(\overline{V}_u, d_u, Q_u)$ passes in the public key verification. Here, $\overline{V}_u = V_u + Q_{ca} + (-Q_{\overline{ca}})$, and $-Q_{\overline{ca}} = (\overline{x}, -\overline{y} \bmod n)$. This is because in the scheme [5], $\overline{\text{CA}}$ had issued the public key Q_u to u if and only if its implicit certificate satisfied the equation $Q_u = \overline{V}_u + Q_{\overline{ca}}$. It is easy to see that the equation holds under the condition that $Q_u = V_u + Q_{ca}$. This means that the forged implicit certificate under the $\overline{\text{CA}}$ is valid to u. Therefore, the implicit certificate scheme in P-HIP [5] is insecure.

In addition, we deplore implicit certificate schemes in [5, 6] lack of a revocation mechanism. Thus, there is a risk that a malicious attacker might try to use the relevant credential that it is no longer valid while the credential itself has not expired. Instead of CRLs and OCSP, we propose a certificate revocation mechanism that uses only one Z_p-symbol as the authentication information to achieve the revocation validation of implicit certificates.

2.3 Related work

Public-key validation is emerged as a popular tool in IoT applications. In PKI, the CA issues and manages public keys of users by using digital certificates. In the traditional explicit certification model,a user's digital certificate $cert_u = (meta, Q_u, sig_u)$ is issued by a trusted CA. The signature sig_u on $cert_u$ implies that the owner of $cert_u$ knows the private key d_u of public key Q_u. In the implicit certification model, the key pair (d_u, Q_u) is computed by the user u in collaboration with the CA. Implicit certificates were introduced in the work of Günther [7] and Girault [8]. Brown et al. [9] defined a general notion of security for implicit certificates, and proved that optimal mail certificates were secure under this definition. However, it has various drawbacks in terms of security and efficiency. In 2013, Campagna [10] presented an implicit certification solution in the Elliptic Curve Qu-Vanstone (ECQV) protocol. Unfortunately, this approach suffered from certificate misbinding attacks. Recently, Barreto et al. [6] proposed an improvement for its security weaknesses and computational efficiency.

The authenticated key establishment between two IoT devices was achieved via HIP [11]. Figure 1 shows that the host identifiers (public keys and HITs) were validated by the HIP peers exchanging X.509 certificates. However, the size of the certificate is much larger than both that of its public key (see Table 1) and the maximum transmission unit of the IEEE 802.15.4 link [12] in IoT networks. Recently, Hossain and Hasan [5] proposed P-HIP in which the ECQV implicit certification scheme was able to reduce the public-key validation data for mutual authentication while protecting the user privacy. In this work,we shows that the ECQV implicit certificate in P-HIP suffered from a forgery attack, that is, a malicious user holding the implicit certificate issued by a CA was able to forge an implicit certificate issued by another CA. Then, a new scheme SEIC is proposed to resist the forge attacks.

3 Preliminaries

In this section, we introduce some notations and Nyberg's one-way accumulator needed later.

3.1 Notations

We shall use the following notations throughout the paper. A set with integers $1, 2, \cdots, n-1$, is written either $\mathbb{Z}_n^*$ or simply $[n-1]$. We denote by $|x|$ the length of the binary string corresponding to x, and $\lceil x \rceil$ the least integer that is greater than or equal to the given number x. Let $\mathbb{F}_q$ be a finite fields, $\mathbb{Z}_n$ be a addition group, and $\mathbb{Z}_n^* = \mathbb{Z}_q \setminus \{0\}$, where q and n are two prims, $q \geq n+1$, and n is the size of a signature (see step 24 in Figure 3 and 5). We let $H : \{0,1\}^r \times \{0,1\}^* \to \{0,1\}^r$ denote a NOWA for one-way hash function, $Hash : \{0,1\}^* \to \mathbb{Z}_n$ and $h : \{0,1\}^* \to \{0,1\}^{rd}$ be two one-way hash functions, where h is used to construct the required H. Let p is a prime number satisfying r=$|p|$, where $A_{\sqcup} \in \mathbb{Z}_p$ (see Section 4.2)).

3.2 Nyberg's one-way accumulator

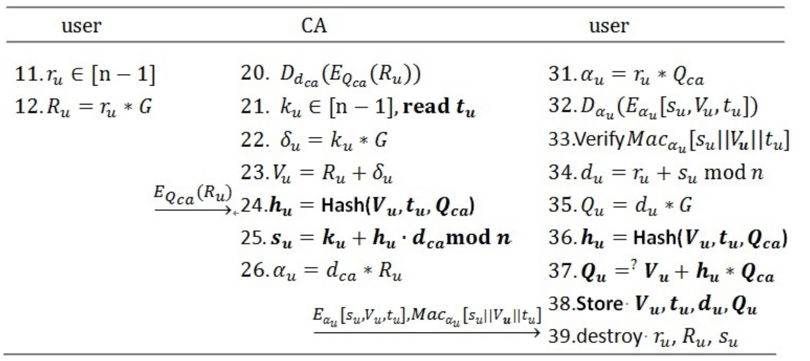
Here, the concept of the Nyberg One-Way Accumulator (NOWA) in [13] is reviewed. Let $H(\cdot,\cdot)$ denote NOWA from $\{0,1\}^r \times \{0,1\}^*$ to $\{0,1\}^r$, and $\odot$ be the bitwise operation AND. The NOWA was constructed by a one-way hash function $h : \{0,1\}^* \to \{0,1\}^{rd}$. Here, $N = 2^d$ is an upper-bound to the number of items to be accumulated, and $r = |q|$ is an integer. All that was required to specify an NOWA was hashing process and AND operation.

Let $h_1, h_2, \cdots, h_n$, $n \leq N$ be the items to be accumulated, and $h(h_i) = y_i$, $i = 1, \cdots, m$ be their hash values. Each hash value is a string of length rd bits. The heart of NOWA was the hashing process. The hashing process applied a hash function h to the input to produce a r-bit output. The hashing process was composed of the following operations: (1)Hashing operation: Hash accumulated item h_i of the input and output a rd bits binary string v_i=$h(h_i)$. (2)Transfer α: NOWA did a transfer operation on the binary string v_i which was divided into r blocks, $(v_{i,1}, \cdots, v_{i,r})$, of length d. The transfer of a block from a d-bit input to a bit output was performed as follows: If $v_{i,j}$ was a string of zero bits, it was replaced by 0; otherwise, $v_{i,j}$ was replaced by 1. That is, $\alpha(v_i)$ =$(b_{i,1}, \cdots, b_{i,r})$, where $b_{i,j} \in \{0,1\}$, j=$1, \cdots, r$. In this way, we can transfer an accumulated item h_i to a bit string, b_i=$\alpha(h(h_i)) \in \{0,1\}^r$, which can be considered as a value of r independent binary random variable if h is an ideal hash function.

In practice, the NOWA is effectively implemented by using the generic symmetry-based hash function and simple bit-wise operations. The NOWA on an accumulated item $h_i \in S$ with an accumulated key $k \in \{0,1\}^r$ was able to be implemented using the AND operation described as $H(k, h_i) = k \odot \alpha(v_i) = k \odot \alpha(h(h_i))$. And it also could be represented as $A = H(k, h_i) = k \odot \alpha(v_i) = k \odot \alpha(h(h_i))$ $(i \in [n])$ if S was a set of accumulated items $S = \{s_1, s_2, \cdots, s_n\}$. $H(\cdot,\cdot)$ has the following properties: (1) Quasi-commutativity: $H(H(k, h_1), h_2) = H(H\ (k, h_2), h_1)$. (2) Absorbency: $H(H(k, h_i), h_i) = k \odot \alpha(h(h_i)) = H\ (k, h_i)$. (3) An item h_i within the accumulated value A can be verified by $H(A, h_i) = A \odot \alpha(h(h_i)) = A$.

4 The proposed scheme

In the section, we propose a secure and efficient implicit certificate (SEIC) scheme to overcome the P-HIP's weaknesses in Section 2.1. In order to make the signature s_u prevent the forgery attack, the SEIC scheme constructs a secure digital signature algorithm by hashing the public key validation data V_u, the time-stamp t_u and the CA's public key Q_{ca}. The scheme also presents a table-and-optimality-based technique that makes the fixed-point multiplication in [6] more computationally efficient. Then, a certificate revocation mechanism is proposed. Finally, a formal proof for the security of SEIC is provided.



user	CA	user
11. $r_u \in [n-1]$	20. $D_{d_{ca}}(E_{Q_{ca}}(R_u))$	31. $\alpha_u = r_u * Q_{ca}$
12. $R_u = r_u * G$	21. $k_u \in [n-1]$, **read** $\boldsymbol{t_u}$	32. $D_{\alpha_u}(E_{\alpha_u}[s_u, V_u, t_u])$
	22. $\delta_u = k_u * G$	33. Verify $Mac_{\alpha_u}[s_u \| \| V_u \| \| t_u]$
	23. $V_u = R_u + \delta_u$	34. $d_u = r_u + s_u \bmod n$
$\xrightarrow{E_{Q_{ca}}(R_u)}$	24. $\boldsymbol{h_u} =$ **Hash**$(\boldsymbol{V_u, t_u, Q_{ca}})$	35. $Q_u = d_u * G$
	25. $\boldsymbol{s_u = k_u + h_u \cdot d_{ca} \bmod n}$	36. $\boldsymbol{h_u} =$ Hash$(\boldsymbol{V_u, t_u, Q_{ca}})$
	26. $\alpha_u = d_{ca} * R_u$	37. $\boldsymbol{Q_u =^? V_u + h_u * Q_{ca}}$
	$\xrightarrow{E_{\alpha_u}[s_u, V_u, t_u], Mac_{\alpha_u}[s_u \| \| V_u \| \| t_u]}$	38. **Store** $\boldsymbol{V_u, t_u, d_u, Q_u}$
		39. destroy r_u, R_u, s_u

Fig. 3. The proposed secure and efficient implicit certificate (SEIC) scheme.

4.1 Proposed SEIC

The user sends a request R_u to the CA via a secure way (i.e.,public key encryption), by choosing a random integer $r_u \in [n-1]$ and then calculating $R_u = r_u * G$. Upon receiving the request, the CA obtains R_u. Then, by picking a random integer $k_u \in [n-1]$ and computing $V_u = R_u + k_u * G$ and s_u=$k_u + h_u \times d_{ca} \bmod n$, the CA issues a public key construction data s_u and a unique public-key validation data V_u to the user, where h_u=$Hash(V_u, t_u, Q_{ca})$, t_u is the current time-stamp of the CA. Before sending (s_u, V_u, t_u) to the user, the CA uses the shared session key, $\alpha_u = d_{ca} * R_u$, to encrypt them and compute their Message Authentication Code (MAC). Upon receiving the messages from CA, the user computes α_u to decrypt $E_{\alpha_u}[s_u, V_u, t_u]$, and verifies $MAC_{\alpha_u}[s_u||V_u||t_u]$. Then, the user generates a private key $d_u = r_u + s_u \bmod n$ and public key $Q_u = d_u * G$. The user validates the result using the equality Q_u=$V_u + Hash(V_u, t_u, Q_{ca}) * Q_{ca}$. The details of the proposed implicit certificate protocol are shown in Figure 3.

The proposed SEIC can prevent the forged attacks in Section 2.1. Since $Hash(\cdot)$ is one-way and collision-resistant, it is hard to compute a pre-image of a given value.

That is, given a randomly chosen $h_u = Hash(V_u, t_u, Q_{ca})$, it is computationally infeasible to find a tuple $(\overline{V}_u, t_u, Q_{\overline{ca}})$ such that $Hash(\overline{V}_u, t_u, Q_{\overline{ca}}) = h_u$. Thus, based on (V_u, t_u, Q_{ca}), it is hard to forge an $\overline{V}_u$ satisfying Q_u=$\overline{V}_u + h_u * Q_{\overline{ca}}$. In other word, in the proposed SEIC, it is computationally infeasible for a malicious user holding the implicit certificate issued by the CA to forge an implicit certificate issued by another CA.

4.2 Performance considerations

Assume that all the users know the system parameter pas=$\{\mathcal{G}, G, n, Hash(\cdot)\}$. The proposed SEIC can be very efficient since it allows a certain amount of precomputation.

Precomputation for $\boldsymbol{h_u * Q_{ca}}$ Assume that b_m (bits) is the memory size used to the precomputation for $h_u * Q_{ca}$. Notice that the binary length of the output of $Hash$ is $|n|$, and the elliptic curve is on the finite field $\mathbb{F}_q$. We observe that the computational efficiency of $\hat{h} * Q_{ca}$ is significantly improved when $\hat{h}$ is restricted to a sufficiently small range. Note that the CA's public key Q_{ca} is commonly a fixed point for each user. Therefore, $h_u * Q_{ca}$ is amenable to optimization methods typical of fixed-point EC multiplications [6]. For a larger integer $h_u \in \{0,1\}^{|n|}$, we select a suitable base B and obtain its expansion (1) on the base B, so that each term $(c_l \cdot B^l) * Q_{ca}$ can be calculated efficiently.

$$h_u = c_\kappa \cdot B^\kappa + c_{\kappa-1} \cdot B^{\kappa-1} + \cdots + c_1 \cdot B + c_0. \tag{1}$$

Here, $0 \le c_l < B$, $l = 0, 1, \cdots, \kappa$, and $\kappa = \lceil |n|/(log_2 B)) \rceil$ is the number of substrings of length $log_2 B$ in h_u.

Given b_m and r, we design a table-and-optimality-based technique: how to choose an optimal base B such that the operation $h * Q_{ca}$ is accelerated. The specific operations are as follows:

(1) Define allowed values $AV = \{B = 2^\theta : 2|q|\lceil |n|/\theta \rceil 2^\theta < b_M\}$. This is because that there are $\kappa \cdot B$ intermediate results $(c_l \cdot B^l) * Q_{ca}$ to be stored, and $x, y \in \mathbb{F}_q$ for a point Q_{ca}=(x, y).
(2) It is recommended to select the largest $B = 2^\theta$ in AV. We notice that $\kappa = \lceil |n|/\theta \rceil$ decreases as $B = 2^\theta$ increases.
(3) By pre-computing $\hat{T}[l][c_l] = (c_l \cdot B^l) * Q_{ca}$ $(0 \le c_l < B, 0 \le l < \kappa)$and then storing them in the memory of the device, the $h_u * Q_{ca}$ operation can be implemented via table look-ups as follows:

$$\begin{aligned}
& h_u * Q_{ca} \\
& = (c_{\kappa-1} \cdot B^{\kappa-1} + \cdots + c_1 \cdot B + c_0) * Q_{ca} \\
& = (c_{\kappa-1} \cdot B^{\kappa-1}) * Q_{ca} + \cdots + (c_1 \cdot B^1) * Q_{ca} + c_0 * Q_{ca} \\
& = \hat{T}[\kappa-1][c_{\kappa-1}] + \cdots + \hat{T}[1][c_1] + \hat{T}[0][c_0].
\end{aligned}$$

This means that $h_u * Q_{ca}$ operation can be attained through $(\kappa - 1)$ point additions.

For example, assuming that $|n|$=256, $|q|$=512, and s_m=512KBs=4194304 bits (IoT devices have a few megabytes of memory (8–32 KB of RAM and 48–512 KB of ROM), e.g., eZ1-Mote [14] has 32 KB of RAM and 512 KB of ROM). In the case, the allowed values is $AV = \{8, 16, 32, 64\}$. We choose B=64, and then κ=$\lceil 256/(log_2 64)\rceil$=43. Ignoring the (usually small) cost of table look-ups, this approach would take only 42 point additions. The size of the memory block storing the intermediate results is $43 \times 64 \times (512 + 512)$ bits =344 KBs.

In comparison, the method in SIMPL [6] using B=16 would require 63 point additions and 128 KBs memory. Thus, the table-and-basis-based technique is expected to be about 1.5 times faster than the method [6].

Implicit certificate revocation Let $HIP^{(\sqcup)}$ be the set of revoked implicit certificates in time slot $\sqcup$. Based on the NOWA H, the revocation manager (RM) compute a NOWA value in $\mathbb{Z}_p$ by accumulating the hash values of implicit certificates in $HIP^{(\sqcup)}$. Then, the RM distributes the value in $\mathbb{Z}_p$ to all users in advance. Keeping just one $\mathbb{Z}_p$-symbol for the revocation verification reduces the storage and communication costs of each user. Specifically, when a user u requests to revoke her/his implicit certificate, the RM can revoke the implicit certificate as follows.

(1) The user u sends the implicit certificate (V_u, t_u, d_u, Q_u) and the CA's ID to the RM. The RM first determines the implicit certificate to be unexpired and correct by using the steps 36-37 in Figure 3.
(2) The RM revokes the unexpired and correct implicit certificate by updating the previous $A_\sqcup$ with the new $A_{\sqcup'}$ for the time epoch $\sqcup'$, where $A_{\sqcup'}$ = $H(A_\sqcup, \text{HIT}_u)$, and HIT_u = $Hash(Q_u)$. The RM then sends the value $A_{\sqcup'}$ to all users via the block chain or a tamper-proof electronic bulletin board.
(3) Each verifier downloads timely the new $A_{\sqcup'}$. The verifier then checks if $H(A_{\sqcup'}, \text{HIT}_u) \neq A_{\sqcup'}$ for the valid public key Q_u, where HIT_u = $Hash(Q_u)$. If the inequality holds, (V_u, t_u, Q_u) is valid; otherwise, it has been revoked.

4.3 Security analysis

Under the assumption that the elliptic curve discrete logarithm problem (ECDLP) is hard on $\mathcal{G}$, we provide the security proof of SEIC as follows.

As shown in Figure 3, the corresponding digital signature scheme DS=$(Gen, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is defined as follows:

- $pas = \{\mathcal{G}, G, n, Hash(\cdot), H\} \leftarrow Gen(1^\kappa)$: On inputting the security parameter κ, the probabilistic algorithm Gen outputs an array of system parameters pas.
- $(d_{ca}, Q_{ca}) \leftarrow \mathcal{K}(pas)$: On inputting the system parameters pas, the probabilistic algorithm $\mathcal{K}$ generates a pair of public and private keys (d_{ca}, Q_{ca}) for a CA.
- $(R_u, (V_u, t_u, s_u)) \leftarrow \mathcal{S}(d_{ca}, R_u)$: On inputting a private key d_{ca} and a message $R_u \in< G >$, the CA runs the probabilistic algorithm $\mathcal{S}$ to produce a signature σ = (V_u, t_u, s_u).

– $\{0,1\} \leftarrow \mathcal{V}(Q_{ca}, R_u, \sigma)$: On inputting the CA's public key Q_{ca}, a message R_u and a signature σ, anyone can run the deterministic algorithm $\mathcal{V}$ to check whether σ is a valid signature. That is, σ is a valid signature if $Q_u = V_u + Hash(V_u, t_u, Q_{ca}) * Q_{ca}$, where $Q_u = R_u + s_u * G$.

The Lemma 1 in Appendix proves the unforgeability of DS in the proposed SEIC against adaptive chosen-message attacks.

Assume a scenario with n_{usr} legitimate users, denoted usr_i for $1 \leq i \leq n_{usr}$, and with n_{ca} CAs, denoted CA_j for $1 \leq j \leq n_{ca}$. Let $(R_i = r_i * G, j)$ denote usr_i's implicit certificate request for CA_j, and let (V_i, t_i, s_i) be the response sent by that CA. Also, let Q_i and d_i denote, respectively, the public and the private keys reconstructed by usr_i from CA_j's response, using that CA's public key Q_j. There are no restrictions on the number of credential requests that can be sent by usr_i to CA_j.

Definition 1 *A (τ', ϵ)-adversary $\mathcal{A}$ (of an implicit certificate scheme) is a probabilistic Turing machine that runs in time at most τ, interacting with legitimate users and CAs by performing each of the following operations any number of times:*

(1) receive a request $(R_i = r_i * G, j)$ from usr_i for an implicit certificate from CA_j; and

(2) send a request $(R_{i'} = r_{i'} * G, j')$ to $CA_{j'}$, and receive response $(s_{i'}, V_{i'}, mac_{i'})$ from $CA_{j'}$.

With probability at least ϵ, $\mathcal{A}$ outputs a triple (r, V, t, s) such that d=$r + s$ is the private key associated with the public key Q reconstructed from V and some Q_z (that is, $d * G$ = $V + Hash(V, t, Q_z) * Q_z$) such that either

(1) [Forgery attack against CA_z]: (V, t, s) was never part of a response of CA_z for the request $(r * G, z)$; or

(2) [Key compromise against usr_i]: (V, t, s) was included in a response of CA_j to some request $(r * G, j)$ originally from usr_i, where $j \neq z$.

A (τ', ϵ)-adversary is considered successful if ϵ is non-negligible for a polynomial time τ'.

In summary, as shown in Figure 4, this model covers a scenario where the adversary $\mathcal{A}$ acts as proxy for requests from users and responses from CAs. Hence, $\mathcal{A}$ can: simply relay the request to the correct CA; modify the value of $R_i = r_i * G$ in the request; modify the user identifier i in the request, thus affecting the value of V in the credential; and/or forward the request to a different CA.

Under the security model of Definition 1(see Figure 4) and the random oracle model, Theorem 2 in Appendix proves the security of the proposed SEIC.

5 Application to HIP in IoT

Public key validation can ensure the authenticity of an HIT in HIP. HIP is based on the Diffie-Hellman key exchange, using public key identifiers from a new host identity name-space for mutual peer authentication. The device uses a 128-bits hash of the public key as HIT.

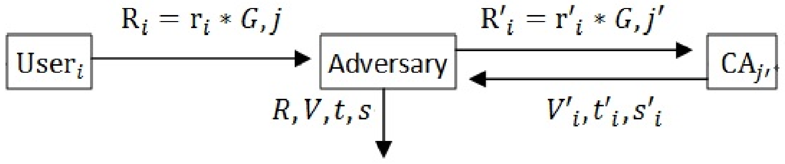


Fig. 4. Security model for implicit certificate adapted from [[9]].

An important challenge in HIP environments is to build a privacy-preserving HIP where authorized devices cannot be tracked, either by eavesdroppers or by the system itself [5]. One common approach for this issue is to provide a IoT device with multiple short-lived public keys. Then, IoT devices can avoid tracking by changing the public keys employed to sign its messages while it move from one network to another one. Hence, messages broadcast from different locations and using distinct public keys cannot be easily linked to any given IoT device. However, the total number of public keys valid simultaneously should be limited [6]. Among the existing solutions, the Secure Credential Management System (SCMS)[15] is one of the most relevant. The approach in SCMS combines privacy and scalability in the so-called butterfly key expansion process. Essentially, this process can issue multiple implicit certificates with a single request from a user. Furthermore, in the proposed SEIC, it reduces the amount of data exchanged and also the number of operations performed by the user.

In this section, we improve SEIC via the butterfly key expansion process, and then construct an improved P-HIP, We also formally analyze the privacy of the improved P-HIP.

5.1 Performance improvement for SEIC

The implicit certificate issuance and revocation in the improved SEIC involves mainly four entities: User, Registration Authority (RA), CA and RM. Assume that there is no CA-RA collusion. They are respectively responsible for the following operations:

- User: the entity that requests credentials from a registration authority (RA). For better efficiency, each request leads to the provisioning of a batch containing β implicit certificates.
- RA: the entity that creates β implicit certificate requests to the CA from a single request of a user (called butterfly key expansion process). Those requests are individually forwarded to the CA, in such a manner that requests associated to different users are shuffled together.
- Credential Authority (CA): responsible for issuing credentials upon the requests by the RA. The credentials are then individually signed and encrypted by the CA before being sent back to the RA, from which they are delivered to the requesting user.
- Revocation Manager (RM): the entity that identifies the implicit certificates of users/devices and, whenever necessary, revokes them by accumulating their hash

values to generate a value $A_{\sqcup}$ in $\mathbb{Z}_p$ (see Section 4.2). Then, the RM distributes timely $A_{\sqcup}$ to all users in advance.

In the improved P-HIP, Figure 5 presents the message exchange in the implicit certificate issuance phase, where the revocation operations are the same as that of the SEIC in Section 4.2). All communications are made via secure ways, using standard protocols (e.g., Transport Layer Security-TLS) or public key encryption.

The user first sends (R_u, f) to the RA. In response to the request of the user, the RA expands the point R_u into β points $\hat{R}_u^{(l)} = R_u + f(l) * G$. Note that f is shared only between the user and the RA. The RA then sends each individual $\hat{R}_u^{(l)}$ to the CA, while shuffling together requests from different batches to ensure their unlinkability.

The CA, in turn, randomizes $\hat{R}_u^{(l)}$ by picking randomly an integer $k_u^{(l)}$ and adding $k_u^{(l)} * G$ to it. The randomized point is used as the butterfly public key validation data $V_u^{(l)}$. Then, according to the procedure in Sections 4.1, the CA generates the hash value for $(V_u^{(l)}, t_u^{(l)}, Q_{ca})$, and outputs its signature $s_u^{(l)}$. The resulting credential is encrypted and verified with the Diffie-Hellman-key $\alpha_u^{(l)} = d_{ca} * \hat{R}_u^{(l)}$ and sent back to the RA. The RA, unable to decrypt the CA's response $\overline{pkg}$, forwards it back to the requesting user, in batch.

Finally, the user computes $\alpha_u^{(l)}$ to decrypt $\overline{pkg}$. It then verifies that the retrieved credential is indeed valid via $Mac_{\alpha_u^{(l)}}[s_u^{(l)}||V_u^{(l)}]$ and $Q_u^{(l)} = V_u^{(l)} + h_u^{(l)} Q_{ca}^{(l)}$, aiming to ensure there is no Man-in-the-Middle attack by the RA. If the verification is successful, the obtained keys, $(V_u^{(l)}, t_u^{(l)}, d_u^{(l)}, Q_u^{(l)})$, can be used for signing messages.

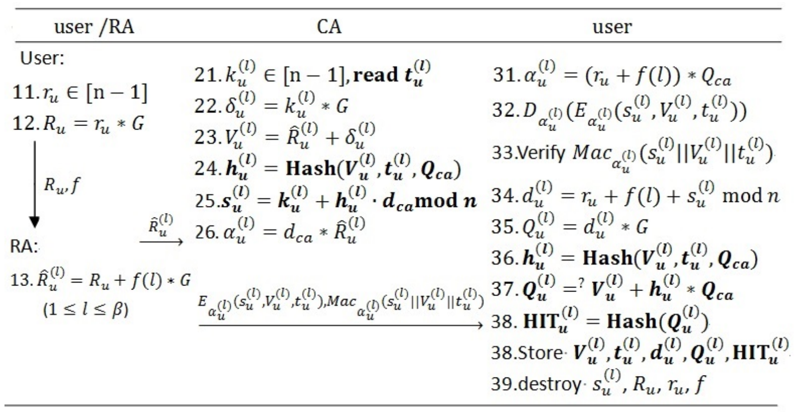

user /RA
CA
user
User:
11. $r_u \in [n-1]$
12. $R_u = r_u * G$
R_u, f
RA:
13. $\hat{R}_u^{(l)} = R_u + f(l) * G$
$(1 \leq l \leq \beta)$
$\hat{R}_u^{(l)}$
21. $k_u^{(l)} \in [n-1]$, **read** $\boldsymbol{t_u^{(l)}}$
22. $\delta_u^{(l)} = k_u^{(l)} * G$
23. $V_u^{(l)} = \hat{R}_u^{(l)} + \delta_u^{(l)}$
24. $\boldsymbol{h_u^{(l)}} = \mathbf{Hash}(\boldsymbol{V_u^{(l)}}, \boldsymbol{t_u^{(l)}}, \boldsymbol{Q_{ca}})$
25. $\boldsymbol{s_u^{(l)}} = \boldsymbol{k_u^{(l)}} + \boldsymbol{h_u^{(l)}} \cdot \boldsymbol{d_{ca}} \mathbf{mod}\ \boldsymbol{n}$
26. $\alpha_u^{(l)} = d_{ca} * \hat{R}_u^{(l)}$
$E_{\alpha_u^{(l)}}(s_u^{(l)}, V_u^{(l)}, t_u^{(l)}), Mac_{\alpha_u^{(l)}}(s_u^{(l)}||V_u^{(l)}||t_u^{(l)})$
31. $\alpha_u^{(l)} = (r_u + f(l)) * Q_{ca}$
32. $D_{\alpha_u^{(l)}}(E_{\alpha_u^{(l)}}(s_u^{(l)}, V_u^{(l)}, t_u^{(l)}))$
33. Verify $Mac_{\alpha_u^{(l)}}(s_u^{(l)}||V_u^{(l)}||t_u^{(l)})$
34. $d_u^{(l)} = r_u + f(l) + s_u^{(l)} \bmod n$
35. $Q_u^{(l)} = d_u^{(l)} * G$
36. $\boldsymbol{h_u^{(l)}} = \mathbf{Hash}(\boldsymbol{V_u^{(l)}}, \boldsymbol{t_u^{(l)}}, \boldsymbol{Q_{ca}})$
37. $\boldsymbol{Q_u^{(l)}} =^? \boldsymbol{V_u^{(l)}} + \boldsymbol{h_u^{(l)}} * \boldsymbol{Q_{ca}}$
38. $\mathbf{HIT}_u^{(l)} = \mathbf{Hash}(\boldsymbol{Q_u^{(l)}})$
38. Store $\boldsymbol{V_u^{(l)}}, \boldsymbol{t_u^{(l)}}, \boldsymbol{d_u^{(l)}}, \boldsymbol{Q_u^{(l)}}, \mathbf{HIT}_u^{(l)}$
39. destroy $s_u^{(l)}, R_u, r_u, f$

Fig. 5. The implicit certificate issuance phase in the improved P-HIP.

Algorithm 1 : Public kry validation

$\{0,1\} \leftarrow PubKeyValid(pas, Q_{ca}, A_{\sqcup}, (V_u, t_u, Q_u, HIP_u))$: It takes pas,Q_{ca}, $A_{\sqcup}$ and (V_u, t_u, Q_u, HIP_u) as the input, and outputs 1 if (Q_u, HIP_u) is authentic or 0 if either the public-key or HIT is forged.

(1) t_u is unexpired.

(2) Q_u is unrevoked by computing $HIT_u^{(l)} = Hash(Q_u)$ and then check if $H(A_{\sqcup}, HIT_u^{(l)}) \neq A_{\sqcup}$.

(3) Q_u is correct and issued by CA if $Q_u^{(l)}$=$V_u^{(l)} + h_u^{(l)} * Q_{ca}$ holds.

(4) If the above checks are true, it returns 1; otherwise, it return 0.

Algorithm 2 : Keying material generation

$null$ or $((d_{pu}, Q_{pu}), V_{pu}) \leftarrow KeyMatGen(HIT_u, d_u, PuzSol)$: It takes the initiator/responder host identifier HIT_u, her/his private key d_u and puzzle/solution $PuzSol$ as the input, and outputs $null$ or a message $((d_{pu}, Q_{pu}), V_{pu})$ as follows: If HIT_u equals $null$ or HIT_u is revoked or $PuzSol$ equals $null$, it returns $null$; otherwise, it returns $((d_{pu}, Q_{pu}), V_{pu})$ by the following operations.

(1) Choose a random integer $r_{pu} \in [n-1]$ and generate a random point V_{pu} = $r_{pu} * G$.

(2) Compute the hash value h_{pu} = $Hash(HIT_u || PuzSol || V_{pu})$.

(3) Use the private key d_u to generate V_{pu}'s validation data Q_{pu} = $d_{pu} * G$, where d_{pu}=$h_{pu} \times d_u + r_{pu} \mod n$.

Algorithm 3 : Keying material validation

$null$ or $\{0,1\} \leftarrow KeyMatValid(HIT_u, Q_u, PuzSol, Q_{pu}, V_{pu})$: It takes the initiator/responder host identifier HIT_u, her/his public key Q_u, puzzle/solution $PuzSol$, the random point V_{pu}, and the validation data Q_{pu} as the input, and outputs $null$ or 1 if V_{pu} is authentic or 0 if the random point is forged. Specifically, if HIT_u equals $null$ or $PuzSol$ equals $null$, it returns $null$; otherwise, it returns $\{0,1\}$ by the following operations.

(1) Compute the hash value h_{pu} = $Hash(HIT_u || PuzSol || V_{pu})$.

(2) Generate the point Q'_{pu} = $h_{pu} * Q_u + V_{pu}$.

(3) If Q'_{pu} = Q_{pu}, it returns 1; otherwise, it returns 0.

Fig. 6. The algorithms in the improved HIP

5.2 Performance improvement for P-HIP

Host identity and host identity tag Suppose that $u \in \{i, r\}$ is an authorized user as an initiator i or a responder r. Then, β implicit certificates $\{V_u^{(l)}, t_u^{(l)}, d_u^{(l)}, Q_u^{(l)}\}$ are first obtained via the process in Section 5.1, l=1, $\cdots, \beta$. For each implicit certificate $\{V_u^{(l)}, t_u^{(l)}, d_u^{(l)}, Q_u^{(l)}\}$, its host identity is the public key $Q_u^{(l)}$. The corresponding HIT can be computed by the user u as HIP$_u^{(l)}$ = $Hash(Q_u^{(l)})$. Hence, the mobile user u can use a new host identity $Q_u^{(l)}$ and its HIP$_u^{(l)}$ to avoid identity tracking when she/he moves from one network to another network.

Host identity validation Here, we present the procedure to validate host identifiers, such as public keys and HITs. Without loss of generality, assume that a prover is the user u who holds an implicit certificate $\{V_u^{(l)}, t_u^{(l)}, d_u^{(l)}, Q_u^{(l)}\}$ and HIP$_u^{(l)}$. Now, the prover provides its host identity $Q_u^{(l)}$, HIP$_u^{(l)}$, the public-key validation data $(V_u^{(l)}, t_u^{(l)})$ and the CA's identity to a verifier. The verifier validates u's implicit certificate by running Algorithm 1 in Figure 6 as $PubKeyValid(pas, Q_{ca}, A_{\sqcup}, (V_u, t_u, Q_u, HIP_u))$.

The correctness of the host identity $Q_u^{(l)}$ can be seen as follows. $Q_u^{(l)}$ = $d_u^{(l)} * G =$ $(r_u + f(l) + s_u^{(l)}) * G$ = $(r_u + f(l) + k_u^{(l)} + h_u^{(l)} \cdot d_{ca}) * G$ =$\hat{R}_u^{(l)} + \delta^{(l)} + (h_u^{(l)} \cdot d_{ca}) * G$ =$V_u^{(l)} + h_u^{(l)} * Q_{ca}$.

Mutual Authentication Let the CA issue $(V_i^{(l)}, t_i^{(l)}, d_i^{(l)}, Q_i^{(l)})$ and $(V_r^{(\iota)}, t_r^{(\iota)}, d_r^{(\iota)}, Q_r^{(\iota)})$ to the initiator and responder, respectively. Assume that CA's public key Q_{ca} and RM's authentication information $A_{\sqcup}$ are correctly sent to all users in advance. The initiator and responder compute HIT$_i^{(l)}$ and HIT$_r^{(\iota)}$ as Section 5.2), respectively. For the improved P-HIP, the mutual authentication procedure is shown in Figure 7.

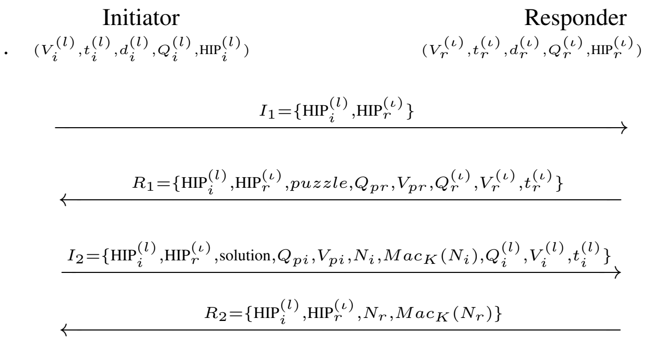
In the improved P-HIP,the operations in the first step and the last step are the same to the common HIP. We will omit the statement of these two steps, and will describe the other intermediate steps and operations in detail.

Upon receiving the message I_1, the responder then creates a message R_1 that contains HIT$_i^{(l)}$, HIT$_r^{(\iota)}$, a puzzle, its keying material (Q_{pr}, V_{pr}), its host identity $Q_r^{(\iota)}$ and the corresponding validation data $(V_r^{(\iota)}, t_r^{(\iota)})$. The keying material is generated by the responder running the Algorithm 2 in Figure 6 as $((d_{pr}, Q_{pr}), V_{pr}) \leftarrow KeyMaterialGen($ $HIT_r^{(\iota)}, d_r^{(\iota)}, puzzle)$.

After receiving the message R_1, the initiator first validates the host identity $Q_r^{(\iota)}$ and HIT$_r^{(\iota)}$ by using the Algorithm 1 in Figure 6 to ensure that the keying material (Q_{pr}, V_{pr}) is generated by an authorized user. Next, the initiator solves the puzzle to get its solution, and then verifies the keying material (Q_{pr}, V_{pr}) in R_1 by running the Algorithm 3 in Figure 6, this is, $\{0, 1\} \leftarrow KeyMatValid(HIT_r^{(\iota)}, Q_r^{(\iota)}, puzzle, Q_{pr}, V_{pr})$. The initiator generates the keying material by using the Algorithm 2 in Figure 6 as $((d_{pi}, Q_{pi}), V_{pi}) \leftarrow KeyMatGen(\ HIT_i^{(l)}, d_i^{(l)}, solution)$. Finally, the initiator computes a session key K=$d_{pi} * Q_{pr}$, and sends the message I_2 to the responder. The message I_2 includes the elements as shown in Figure 7. If one of the above checks returns false, the initiator exits.

Similar to the initiator's approaches, the responder verifies the authenticity of the host identity $Q_r^{(\iota)}$ and $\text{HIT}_r^{(\iota)}$, validates the keying material (Q_{ir}, V_{ir}) in I_2, and authenticates the initiator. Next, the responder computes the session key using d_{pr} and Q_{ir} as $K = d_{pr} * Q_{ir}$, and then validates the $MAC_K(N_i)$.

Finally, at the last step, the responder sends R_2 and the initiator validates $MAC_K(N_r)$ to ensures that a shared session key K is created successfully. After this point, the communications between them are encrypted using K.



Initiator
Responder
$(V_i^{(l)}, t_i^{(l)}, d_i^{(l)}, Q_i^{(l)}, \text{HIP}_i^{(l)})$
$(V_r^{(\iota)}, t_r^{(\iota)}, d_r^{(\iota)}, Q_r^{(\iota)}, \text{HIP}_r^{(\iota)})$
$I_1 = \{\text{HIP}_i^{(l)}, \text{HIP}_r^{(\iota)}\}$
$R_1 = \{\text{HIP}_i^{(l)}, \text{HIP}_r^{(\iota)}, puzzle, Q_{pr}, V_{pr}, Q_r^{(\iota)}, V_r^{(\iota)}, t_r^{(\iota)}\}$
$I_2 = \{\text{HIP}_i^{(l)}, \text{HIP}_r^{(\iota)}, \text{solution}, Q_{pi}, V_{pi}, N_i, Mac_K(N_i), Q_i^{(l)}, V_i^{(l)}, t_i^{(l)}\}$
$R_2 = \{\text{HIP}_i^{(l)}, \text{HIP}_r^{(\iota)}, N_r, Mac_K(N_r)\}$

Fig. 7. The mutual authentication in the improved P-HIP

User privacy In the improved P-HIP, a user may utilize a remaining unused implicit certificate to generate the new host identity while she/he moves to a different network or wants to update its host identity. This way provides a secure interact between different parties without compromising user privacy. Let us suppose, an adversary is provided with the implicit certificates, $\{(V_u^{(l)}, t_u^{(l)}, d_u^{(l)}, Q_u^{(l)})\}_{l=1}^{\varpi}$, of a user u that were used in ϖ number of networks. The goal of the adversary is to infer that the implicit certificates belong the same user. In Section 6.2, we utilize the formal privacy definition [20], to prove that the improved P-HIP gives rigorous, rather than ad-hoc or intuition-based privacy guarantees.

6 The security analysis of the improved P-HIP

In this section, we analyze the security of the improved SEIC and the privacy of improved P-HIP.

6.1 The security analysis for improved SEIC

Here, we show that the improved SEIC in Section 5.1 is secure. Under the attack model in Definition 1, Theorem 2 shows that there is no adversary $\mathcal{A}$ that is successful against the proposed SEIC. Form Lemma 1, we know that SEIC's signature scheme DS is secure against adaptive chosen-message attacks. The improved SCMS in Section is the butterfly key expansion of the proposed SEIC. The attacks in Definition 1 does not invalidate SEIC's security claims for at least three reasons[6]. The first is that in the improved SEIC, one of security assumptions is that there is no CA-RA collusion. Next, SCMS recommends using the ECDSA-signature algorithm [15], for which SEIC's signature scheme DS is a secure ECDSA-signature against the attacks in Definition 1. Finally, the latest version of SCMS already suggests the countermeasure hereby proposed [16], that is, the signer's certificate information is included in the hash computation. Therefore, the improved SEIC remains secure against the forgery in Definition 1.

6.2 The formal privacy analysis of the improved P-HIP

We first define the privacy model, and then formally analyze the privacy of the improved P-HIP.

Privacy model We now consider Ouafi and Phan's privacy model [20]. In this model, attacker $\mathcal{A}$ can eavesdrop on all the channels between two users, and he/she can also perform any active or passive attacks. In this regard, $\mathcal{A}$ needs to model the following queries in polynomial time:

Execute($\mathcal{P}$,$\mathcal{U}, s$): This query represents the passive attacks. In this context, the attacker can eavesdrop all the transmitted messages between the user $\mathcal{U}$ and a party $\mathcal{P} \in \{CA, RA, \mathcal{V}\}$ in the s-th session, where the user $\mathcal{V}$ satisfies $\mathcal{V} \neq \mathcal{U}$. Consequently, the attacker obtains all the exchanged data between $\mathcal{U}$ and $\mathcal{P}$.

Send($\mathcal{U}, \mathcal{V}, m, s$): This query models the active attacks in the system. In this query, attacker $\mathcal{A}$ has the permission to impersonate a user $\mathcal{U}$ in the s-th session, and forwards a message m to another user $\mathcal{V}$. Besides, the attacker has the permission to block the exchanged message m between $\mathcal{U}$ and $\mathcal{V}$.

Query($\mathcal{U}, m_1, m_2$): This query models the adversary's ability to investigate a user. For this, $\mathcal{A}$ sends m_1 to $\mathcal{U}$ and receives m_2 from $\mathcal{U}$.

Corrupt($\mathcal{U}, K$): In this query, the attacker $\mathcal{A}$ has the permission to access secret information K stored in the user $\mathcal{U}$'s memory.

Test($\mathcal{U}_0$,$\mathcal{U}_1, s$): This query is the only query that does not correspond to any of $\mathcal{A}$'s abilities or any real-world event. This query allows to define the indistinguishability-based notion of untraceable privacy.

If the party has accepted and is being asked a Test query, then depending on a randomly chosen bit $b \in \{0, 1\}$, $\mathcal{A}$ is given $\mathcal{U}_b$ from the set $\{\mathcal{U}_0$,$\mathcal{U}_1\}$. Informally, $\mathcal{A}$ succeeds if it can guess the bit b. In order for the notion to be meaningful, a Test session must be fresh in the sense of Definition 3.

Definition 2 (Partnership and session completion) *An initiator instance i and a responder instance r are partners if, and only if, both have output Accept(i) and Accept(r), respectively, signifying the completion of the protocol session.*

Definition 3 (Freshness) *A party instance is fresh at the end of execution if, and only if (1) it has output Accept with or without a partner instance and (2) both the instance and its partner instance (if such a partner exists) have not been sent a Corrupt query.*

Definition 4 (Indistinguishable Privacy (INDPriv)) *It is defined using the game $\mathcal{G}$ played between a malicious adversary $\mathcal{A}$ and a collection of initiators and responders and RAs and CA instances. $\mathcal{A}$ runs the game $\mathcal{G}$ whose setting is as follows.*

- *Learning phase: $\mathcal{A}$ is able to send any Execute, Send, Query, and Corrupt queries and interact with the RA, the CA and users $\mathcal{U}_0$, $\mathcal{U}_1$ that are chosen randomly.*
- *Challenge phase: The attacker selects two users $\mathcal{U}_0$ and $\mathcal{U}_1$, and forwards a Test query $(\mathcal{U}_0, \mathcal{U}_1, s)$ to challenger $\mathcal{C}$. After that, $\mathcal{C}$ randomly selects $b \in \{0,1\}$ and the attacker determines a user $\mathcal{U}_b \in \{\mathcal{U}_0, \mathcal{U}_1\}$ using Execute, Send and Query queries.*
- *Guess phase: The attacker $\mathcal{A}$ finishes the game $\mathcal{G}$ and outputs a bit $\hat{b} \in \{0,1\}$ as guess of b. The success of attacker $\mathcal{A}$ in the game $\mathcal{G}$ and consequently breaking the security of INDPriv is quantified via $\mathcal{A}$'s advantage in recognizing whether attacker $\mathcal{A}$ received $\mathcal{U}_0$ or $\mathcal{U}_1$, and is denoted by $Adv_{\mathcal{A}}^{INDPriv}(k) = |Pr[\hat{b} = b] - 1/2|$, where k is a security parameter.*

Theorem 1 *The improved P-HIP satisfies indistinguishable privacy.*

Proof 1 *In the improved P-HIP, after a successful authentication, the user $\mathcal{U}_0$ update its secret key $d_{\mathcal{U}_0}$. Besides, the host identities $HIP_{\mathcal{U}_0}$ change in each session. Therefore, it will be difficult for an adversary to perform any traceability attack by performing the following phases:*

- *Learning phase: In the μ-th authentication instance, the adversary $\mathcal{A}$ is able to send any Execute$(CA, \mathcal{U}_0, \mu)$ queries and obtains the public key $Q_{\mathcal{U}_0}^{(\mu)}$ and the host identity $HIP_{\mathcal{U}_0}^{(\mu)}$ such that $HIP_{\mathcal{U}_0}^{(\mu)} = Hash(Q_{\mathcal{U}_0}^{(\mu)})$ holds.*
- *Challenge phase: The adversary $\mathcal{A}$ selects two fresh users $\mathcal{U}_0$, $\mathcal{U}_1$ and forwards a Test query $(\mathcal{U}_0, \mathcal{U}_1, \mu+1)$ to the challenger $\mathcal{C}$. Next, according to the randomly chosen bit $b \in \{0,1\}$, $\mathcal{A}$ is given a user $\mathcal{U}_b \in \{\mathcal{U}_0, \mathcal{U}_1\}$. After that, the adversary $\mathcal{A}$ sends a query Execute$(CA, \mathcal{U}_b, \mu+1)$ and obtains the public key $Q_{\mathcal{U}_b}^{(\mu+1)}$ and the host identity $HIP_{\mathcal{U}_b}^{(\mu+1)}$, where $HIP_{\mathcal{U}_b}^{(\mu+1)} = Hash(Q_{\mathcal{U}_b}^{(\mu+1)})$.*
- *Guess phase: In the Learning phase the user $\mathcal{U}_0$ updates its secret $d_{\mathcal{U}_0}$, therefore for the two subsequent sessions μ and $\mu+1$, the public keys $Q_{\mathcal{U}_0}^{(\mu)} = d_{\mathcal{U}_0}^{(\mu)} * G$ and $Q_{\mathcal{U}_b}^{(\mu+1)} = d_{\mathcal{U}_b}^{(\mu+1)} * G$ are calculated as follows: $d_{\mathcal{U}_0}^{(\mu)} = [(r_{\mathcal{U}_0} + f(\mu)) + k_{\mathcal{U}_0}^{(\mu)} + Hash((r_{\mathcal{U}_0} + f(\mu)) * G, t_{\mathcal{U}_0}^{(\mu)}, Q_{ca})d_{ca}] \mod n$, $d_{\mathcal{U}_b}^{(\mu+1)} = [(r_{\mathcal{U}_b} + f(\mu+1)) + k_{\mathcal{U}_b}^{(\mu+1)} + Hash((r_{\mathcal{U}_b} + f(\mu+1)) * G, t_{\mathcal{U}_b}^{(\mu+1)}, Q_{ca})d_{ca}] \mod n$. Note that $r_{\mathcal{U}_0}$, $k_{\mathcal{U}_0}^{(\mu)}$, $r_{\mathcal{U}_b}$ and $k_{\mathcal{U}_b}^{(\mu+1)}$ are the random numbers. Assume that the hash function $Hash$ is truly random, mapping each data item independently and uniformly to the range $\{0,1\}^r$, that is, $Pr[Hash(x) = Hash(x')] = \frac{1}{2^r}$ where $x \neq x'$. Since $t_{\mathcal{U}_0}^{(\mu)} \neq t_{\mathcal{U}_b}^{(\mu+1)}$, therefore $d_{\mathcal{U}_b}^{(\mu+1)} = d_{\mathcal{U}_b}^{(\mu+1)}$ with the probability less than $\frac{1}{2^{r-1}}$. In other word, $Q_{\mathcal{U}_0}^{(\mu)} = Q_{\mathcal{U}_b}^{(\mu+1)}$ holds with the probability less than $\frac{1}{2^{r-1}}$. Again, $HIP_{\mathcal{U}_0}^{(\mu)} = Hash(Q_{\mathcal{U}_0}^{(\mu)})$ and $HIP_{\mathcal{U}_b}^{(\mu+1)} = Hash(Q_{\mathcal{U}_b}^{(\mu+1)})$. Hence, the adversary needs make a random guess for $HIP_{\mathcal{U}_b}^{(\mu+1)}$. In this context, the advantage of the adversary recognizing $\mathcal{U}_0$ or $\mathcal{U}_1$, can be denoted $Adv_{\mathcal{A}}^{INDPriv}(k) = |Pr[\hat{b} = b] - 1/2| < \epsilon$, where $\epsilon = \frac{1}{2^{r-1}}$ is negligible when r is large enough.*

7 Performance analysis and comparison

In this section, based on the improved scheme in Section 5, we compare it with other similar solutions in the literature in terms of the desired security properties, computation cost and communication cost.

7.1 Performance comparison

A comparison of the security properties among the improved scheme with other implicit certificate schemes [5, 6, 15] is given in Table 2. The improved scheme in Section V is secure against the forger and credential misbinding attacks. The signature algorithm's input (see step 24 Figure 3 and 5) includes both $(V_u^{(l)}, t_u^{(l)})$ and the signer's public key Q_{ca}. In the P-HIP scenario, enforcing this technique when signing public-key validation data can avoid forgery attacks that builds upon the properties of butterfly keys. Under the attack model in Definition and Ouafi and Phan's privacy model, the improved scheme provides the rigorous security proof in Section 4.3 and 6.1 and the formal privacy analysis in Section 6.2, respectively. In addition, the improved scheme achieves the revocation verification of a public key by performing one NOWA operation in Section 4.2. However, the schemes [5, 6, 15] focused on the informal analysis of user privacy, and did not consider the revocation of unexpired implicit certificates. Note that the formal security proof did not provided in [5].

Table 2. Performance comparison based on security properties with respect to implicit certificate schemes

Scheme	SP1	SP2	SP3	SP4	SP5
ECQV [15]	No	formal	informal	No	-
SIMPL [6]	Yes	formal	informal	No	SCMS
P-HIP [5]	No	informal	informal	No	HIP
SEIC	Yes	formal	No	Yes	HIP
improved P-HIP	Yes	formal	formal	Yes	SCMS, HIP
SP1: preventing the credential misbinding attacks and a forgery attack					
SP2: security proof; SP3: user privacy proof;					
SP4:credential revocation; SP5: Compatibility					

7.2 Effectiveness analysis

We evaluate the effectiveness of the improved scheme in terms of the computation and communication costs.

Experimental results To show the effectiveness of the improved scheme with respect to the existing implicit certificate schemes, we conduct simulations of the cryptographic operations used by various schemes on an Intel(R) Core(TM) i7-8550U CPU@1.80

GHz laptop computer with 8.00 GB memory and Windows10 using JDK1.8 (operating as the initiator or the responder as per the scheme). The simulations used the JPBC library jpbc-2.0.0 [18] to evaluate the execution time of different cryptographic operations.

We create an ECC self-signed X.509 certificates using the type A pairings on the curve $y^2 = x^3 + x$ over the finite field $\mathbb{F}_q$. SHA-256 is chosen as the cryptographic hash function $Hash$. In addition, we select SHA-512 for hashing h in NOWA H with a 128 bit output, where $N = 2^4$ is an upper bound to the number of accumulated items. When $N > 2^4$, we do this by selecting $\eta = \lceil N/(2^4) \rceil$ different SHA-512 as Remark 1 in [19]. For the function $Hash$ and the message authentication code (MAC), the SHA-256 is chosen as suggested. Furthermore, the leftmost 128 bits in the output of $Hash(Q_u)$ is taken as a HIT corresponding public key Q_u. With the above parameter settings, we consider the average value of over 100 trials for an operation $o \in \{Hash, H, a(\text{Point addition}), m(\text{Point multiplication}), e(\text{ AES encryption}), d(\text{AES decryption})\}$. The results are as follows: T_{Hash} = 1.2828milliseconds(ms), T_H = 53.8039(ms), T_a = 1.3418(ms), T_m = 96.9339(ms), T_e = 13.1607(ms), and T_d = 3.7243(ms). In particular, the average time performing an addition or a multiplication of two numbers is 0.6626ms or 0.7615ms, which is negligible compared to other operations.

Implicit certificate issuance

Computation cost Computation costs are the principal constraint for IoT users/devices, and we show a reduction in required computation in the improved P-HIP as compared to the existing schemes. Table 3 shows the computation cost of a user in different schemes. The improved P-HIP is similar to the approach discussed in [6], the key difference is that instead of 63 point addition, $h * Q_{ca}$ is computed by using 42 point addition. In both schemes, the user generates the request $(R_u = r_u * G, f)$, and then obtains an implicit certificate the β by computing $\alpha_u^{(l)}$, $d_u^{(l)} * G$ and $h_u^{(l)} * Q_{ca}$. It means that in the improved P-HIP, the computation cost of the user is β(2T$_m$+ 44T$_a$+ T_d + 3T$_{Hash}$)+T$_m$. In addition, the verifier performs only one H operation to check whether the unexpired implicit certificates is unrevoked. From Figure 8 (a), it is evident that the computation cost of a user increases with the number of implicit certificate, but it grows relatively slowly in the improved scheme. In particular, the improved P-HIP makes a user has the smallest computation cost.

Table 3. The cost comparison of different schemes in the implicit certificate issuance phase

Scheme	Computation cost	Communication cost
ECQV[15]	β(3T$_m$+T$_a$ +T$_{Hash}$)+T$_m$	$\geq \beta(2\lvert q\rvert + 800) + 2\lvert q\rvert + \lvert f\rvert$
SIMPL[6]	β(2T$_m$+64T$_a$ +T$_{Hash}$)+T$_m$	$\geq \beta(2\lvert q\rvert + 320) + 2\lvert q\rvert + \lvert f\rvert$
P-HIP [5]	β(3T$_m$+2T$_a$+T$_d$+2T$_{Hash}$)	$\beta(4\lvert q\rvert + 320)$
SEIC	β(3T$_m$+44T$_a$+T$_d$+3T$_{Hash}$)	$\beta(4\lvert q\rvert + 320)$
Improved P-HIP	β(2T$_m$+44T$_a$+T$_d$+3T$_{Hash}$)+T$_m$	$\beta(2\lvert q\rvert + 320) + 2\lvert q\rvert + \lvert f\rvert$

Communication cost The advantage of the improved scheme is that the communication cost is low for IoT users/devices in the implicit certificate issuance. The communication cost comparison of these schemes [5, 6, 15] is shown in Table 3. To give a detailed quantitative analysis, we create a ECC self-signed X.509 certificates using the OpenSSL library [2], and choose n = 160 bits and q = 512 bits. The sizes of an identity and a time-stamp are recommended to be 20 bytes [16]. In the improved P-HIP, the communication cost at the user is as follows: The size of β responses from the CA $(c_u^{(l)}, Mac_u^{(l)})$ are $\beta(2|q| + n + 320)$ bits, and the size of a request is $|R_u| + |f|$=2$|q|$+f bits. The total communication cost of a user is $(2\beta + 2)|q| + 320\beta + |f|$ bits. We notice that the size of $meta$ in [6, 15] is not less than 160 bits since $meta$ contains at least a time-stamp. The ECDSA-based signature outputs at least two numbers in $\mathbb{F}_q$. The length of $sig_u^{(l)}$ in ECQV[15] is $2|q|$. However, the size of $|sig_u^{(l)}|$ is $|q|$ since $sig_u^{(l)}$ is a number in $\mathbb{Z}_n^*$. Figure 8 (a) and (b) shows that the improved scheme makes a user have both the smallest computation cost and the smallest communication cost.

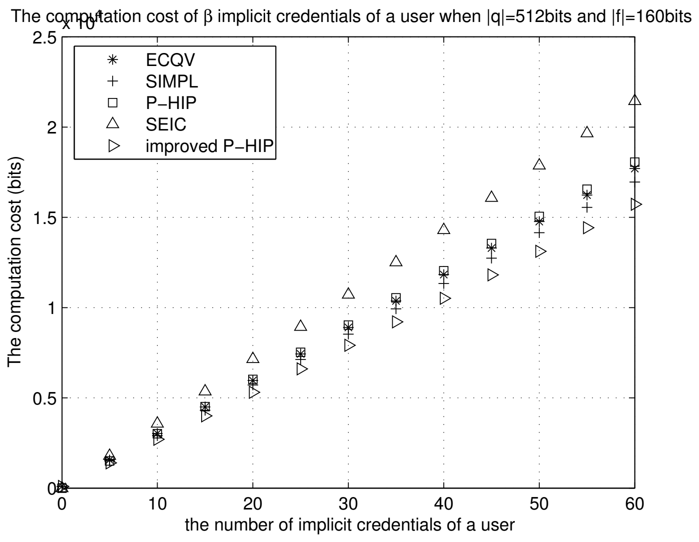
Mutual authentication: Implicit credential validation We also evaluate the benefits of the improved P-HIP when users perform mutual authentication. The gains in this case originate from the following observations.

In Section 5.2), the cost of validating a public key and its HIT is T$_H$+ $\hat{\text{T}}_m$+ T$_a$+2T$_{Hash}$, where $h_u^{(l)} * Q_{ca}$ can be accelerated at cost $\hat{\text{T}}_m$ = 42T$_a$< T$_m$. In addition, the costs for keying material generation and keying material validation (see Algorithm 2 and 3 in Figure 6) are 2T$_m$+ T$_a$+ T$_{Hash}$) and T$_m$+ T$_a$+ T$_{Hash}$, respectively. However, the generation/validation of a MAC requires one hash operation. Therefore, the total computational cost of an initiator or a responder is T$_H$ +3T$_m$+ 40T$_a$+ 6 T$_{Hash}$ ≈ 411.3416ms.

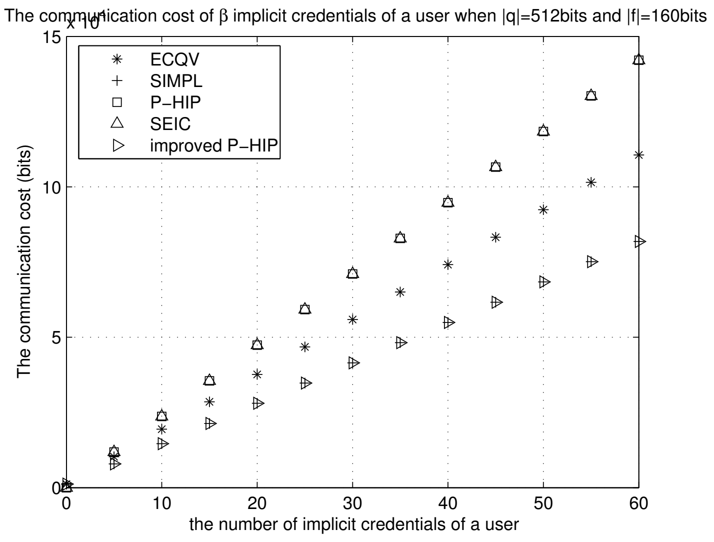
The improved HIP has stronger capabilities (such as public key revocation verification) than P-HIP [5], with an approximate computational cost and the same communication cost. In P-HIP, an initiator or a responder takes the total computational cost to be 4T$_m$+ 3T$_a$+ 6 T$_{Hash}$ $\approx$ 399.4578ms. This value is close to the computational cost of the user in the improved P-HIP. On the other hand, the time for the device to perform a public key revocation verification operation is 53.8039(ms). Since $A_{\sqcup}$ is sent to all users in advance, both schemes have the same communication cost. The result gives a glimpse of SEIC's potential to speed up both signature verification and public-key validation in HIP environments.

8 Conclusion

In this article, we propose SEIC that can improve the security of the P-HIP and the efficiency of EC point multiplications for IoT devices. For fix-point multiplication, the proposed method is about 1.5 times faster than the method in SIMPL scheme. At the same time, by making use of the butterfly key expansion process, we constructs an improved P-HIP by reducing the amount of data exchanged and the number of operations performed by user. Our evaluation shows that the improved P-HIP helps to achieve all the important security properties and ensures the user privacy with reasonable computation cost. However, one limitation is that for the solution to become a reality, the



(a) The computation cost of β implicit certificates of a user



(b) the communication cost of β implicit certificates of a user

Fig. 8. The computation and communication costs of β implicit certificates of a user in the improved P-HIP

revocation verification information $A_{\sqcup}$ must be updated timely and sent to all users in advnce, which can be costly. How to effectively implement the update and release for $A_{\sqcup}$ is a potential future direction of this research.

Appendix

Lemma 1 *Assuming* (1) *ECDLP is hard for* $\mathcal{G}$ *and* (2) $Hash$ *is a random oracle, the Schnorr signature scheme* DS *in Figure 3 is secure against adaptive chosen-message attacks.*

Proof 2 *Let* $Adv^{cma}_{DS,\mathcal{A}}(\kappa)$ *be the probability that* $\mathcal{A}$ *breaks the above Schnorr signature scheme* DS *and achieve a forgery. Assume that* $Adv^{cma}_{DS,\mathcal{A}}(\kappa)$ *is nonnegligible. We will construct an algorithm* $\mathcal{B}$ *which can solve the ECDLP in* $\mathcal{G}$*.*

Let G *be a generator of* $\mathcal{G}$*. Given a point* Q_{ca}=$d_{ca} * G \in < G >$ *as a challenge to* $\mathcal{B}$*, it aims to output such a value* $d_{ca} \in Z_q^*$*. The hash function* $Hash$ *behaves as a random oracle.*

$\mathcal{B}$ *starts* $\mathcal{A}$ *on input* 1^κ*. Let* $T_1(\kappa)$ *denote the bound of the number of CAs.* $\mathcal{B}$ *picks at random a* $i \in \{1, \cdots, T_1(\kappa)\}$*, guessing that* $\mathcal{A}$ *will succeed against the entity* i*.* $\mathcal{B}$ *runs* $\mathcal{K}(pas)$ *to generate for each entity its private/public pair, except* i*.* i *is given a public key* Q_{ca} *, while the corresponding private key* d_{ca} *is unknown to* $\mathcal{B}$*. A* i*'s signature on a message can be generated by querying the signing oracle* $\mathcal{S}(d_{ca}, \cdot)$*.*

$\mathcal{B}$ *can simulate the entity* i *to respond messages to via the following oracles:*

Hash queries: *At any time,* $\mathcal{A}$ *can query* $Hash$*.* $\mathcal{B}$ *maintains a list* H_{list} *of tuples* (R, V, t, k, Q_{ca}) *which is initially empty, and a query counter* μ *which is initially set to* 0*.* $\mathcal{A}$ *provides a new pair* (R, V, t, Q_{ca}) *for hash query by first choosing a message* R *and then computing* $V = R + k * G$*, where* k *is a random number in* Z_n^**. Upon a hash query* (R, V, t, Q_{ca}) *for which there exists a record* (R, V, t, Q_{ca}, h) *in* H_{list}*,* $\mathcal{B}$ *return* h *to* $\mathcal{A}$*; otherwise,* $\mathcal{B}$ *uniformly chooses a random number* $h \in Z_n^*$ *as the value of* $H(V, t, Q_{ca})$*, places* (R, V, t, Q_{ca}, h) *into* H_{list}*, and returns* h *to* $\mathcal{A}$*.*

Signature queries:*Proceeding adaptively, the adversary* $\mathcal{B}$ *answers* $\mathcal{A}$*'s queries for signing oracle* $\mathcal{S}(d_{ca}, \cdot)$*. When* $\mathcal{A}$ *provides a query message* R*,* $\mathcal{B}$ *works as follows:*

(1) Randomly choose two numbers $u, v \in Z_q^*$*, and compute* $V = R + u*G + (-v*Q_{ca})$*;*
(2) Set $s = u$*,* $H(V, t, Q_{ca}) = v$*, and place* (R, V, t, Q_{ca}, v) *into* H_{list}*;*
(3) Returns (v, s) *as a signature on message* R *to* $\mathcal{A}$*.*

After $\mathcal{A}$ *makes* H*-hash query on* (R, V, t, Q_{ca}) *to get* $v = H(V, t, Q_{ca})$*, it can verify that* (V, s) *satisfies* $R + s * G = V + v * Q$*. Therefore,* (V, s) *is a valid signature on message* R *with respect to* j*'s public key* Q_{ca}*. Since* V *and* s *follows the uniform distribution, and* $Hash$ *behaves as a random oracle,* $\mathcal{A}$ *cannot distinguish between* $\mathcal{B}$*'s response and the real life.*

Output: *Eventually, suppose* $\mathcal{A}$ *returns a forgery* (R^*, V^*, t^*, s^*)*, where* (V^*, t^*, s^*) *is a valid forgery distinct from any previously given signature on message* R^* *with respect to the public key* Q_{ca}*.*

According to the above proof, $\mathcal{A}$ *can find a valid signature with non-negligible probability* $Adv^{cma}_{DS,\mathcal{A}}(\kappa)$*. Then, by using the forming lemma,* $\mathcal{A}$ *can outputs a new* *forgery* $(V^*, t^*, \hat{s}^*)$ *on the same message* R^* *and a different oracle* $H\hat{a}sh(\cdot)$, *with non-negligible probability, such that* $Hash(V^*, t^*, Q^*_{ca}) \neq H\hat{a}sh(V^*, t^*, Q^*_{ca})$ *and* $s^* \neq \hat{s}^*$. *From this, we get*

$$s^* = k + d_{ca} Hash(V^*, t^*, Q^*_{ca}) \bmod n$$
$$\hat{s}^* = k + d_{ca} H\hat{a}sh(V^*, t^*, Q^*_{ca}) \bmod n$$

Thus, $\mathcal{B}$ *can solve out the private key*

$$d_{ca} = \frac{s^* - \hat{s}^*}{Hash(V^*, t^*, Q^*_{ca}) - H\hat{a}sh(V^*, t^*, Q^*_{ca})} \bmod n \qquad (2)$$

which is just $\mathcal{B}$*'s challenge* d_{ca}. *The choices of* i *in algorithm* $\mathcal{B}$ *imply that with probability at least* $\frac{1}{T_1(\kappa)}$ *he can 'hit' the correct value of* $\mathcal{A}$. *Thus,* $Adv^{ecdl}_{\mathcal{G},\mathcal{B}}(\kappa) \geq \frac{Adv^{cma}_{DS,\mathcal{A}}(k)}{T_1(\kappa)}$, *where* $Adv^{ecdl}_{\mathcal{G},\mathcal{B}}(\kappa) = Pr[pas \leftarrow Gen(1^\kappa); Q_{ca} \leftarrow \mathcal{G}; d_{ca} \leftarrow \mathcal{B}(pas, Q_{ca}) : d_{ca} * G = Q_{ca}]$. *Since the ECDLP is assumed to be hard in* $\mathcal{G}$, *then* $Adv^{ecdl}_{\mathcal{G},\mathcal{B}}(\kappa)$ *must be negligible. This contradicts the assumption that* $Adv^{cma}_{DS,\mathcal{A}}(\kappa)$ *is nonnegligible. Thus, we conclude that* $Adv^{cma}_{DS,\mathcal{A}}(\kappa)$ *is negligible for all adversaries* $\mathcal{A}$.

Theorem 2 *Assuming* (1) *ECDLP is hard for* $\mathcal{G}$ *and* (2) $Hash$ *is a random oracle, there is no adversary A that is successful against the proposed scheme SEIC.*

Proof 3 *Assume that the proposed scheme SEIC is not secure in the case where the hash function* $Hash$ *is a random oracle. Then there exists a successful* (τ', ϵ)*-adversary* $\mathcal{A}$. *We construct a polynomial-time algorithm* $\mathcal{B}$ *that uses* $\mathcal{A}$ *as a subroutine to compute logarithms in* $\mathcal{G}$ *with non-negligible probability.*

The input to $\mathcal{B}$ *consists of a discrete logarithm challenge* $Q \in_R \mathcal{G}$, $Q \neq \mathcal{O}$, *and the desired output of* $\mathcal{B}$ *is an integer* $d \in [1, n)$ *such that* $Q = d * G$. *We shall construct* $\mathcal{B}$ *in two stages. The first stage* $\mathcal{B}_1$ *takes as input* (Q, R, H_1) *where* R *is a random message, and* H_1 *is a random oracle independent of* $Hash$. $\mathcal{B}_1$ *can use* $\mathcal{A}$ *as a subroutine. The desired output of* $\mathcal{B}_1$ *is either (i) an integer* $d \in [1, n)$ *such that* $Q = d * G$, *or (ii) an ordered pair* (V, t, s) *such that* $s * G + R = V + H_1(V, t, Q) * Q$ *(i.e.,* (V, t, s) *is a signature of message* R *with respect to the public key* Q*). If case (i) occurs, then* $\mathcal{B}$ *outputs* d *and terminates. If case (ii) occurs, then Lemma 1 is used to reduce the signature forger* $\mathcal{B}_1$ *to a discrete logarithm solver in order to extract* d. *If this stage is successful, then* $\mathcal{B}$ *outputs* d *and terminates.*

To find d, *algorithm* $\mathcal{B}_1$ *runs algorithm* $\mathcal{A}$. *Algorithm* $\mathcal{A}$ *expects there to be one or more CAs, each with a public key for which* $\mathcal{A}$ *is not given the private key, and zero or more requester* $user_i$ *making one or more requests* R_i *for which* $\mathcal{A}$ *is not given the discrete logarithm* r_i. *Algorithm* $\mathcal{B}_1$ *randomly selects one of the CA public keys or one of the requests to be the challenge point* Q *which is the input of* $\mathcal{B}$. *The other request points and CA public keys can be selected by* $\mathcal{B}_1$ *according to the normal procedure of selecting a random secret integer and multiplying* G *by this value. Let* τ' *be the total number of CA public keys and requests. We shall see that there will be a* ϵ/τ' *probability that* $\mathcal{A}$ *can be used to obtain* d *or a forgery of a signature with public key* Q.

Since $\mathcal{A}$ *can request an implicit certificate from the CA with public key* Q *(if* $\mathcal{B}_1$ *has selected such a CA) and expect a legitimate response,* $\mathcal{B}_1$ *must supply a response that* *seems legitimate at least from $\mathcal{A}$'s perspective. (Otherwise $\mathcal{A}$ is not guaranteed success, and $\mathcal{B}_1$ may not find $\mathcal{A}$ useful to find d.) However, $\mathcal{B}_1$ does not know the private key d associated with Q. But since $Hash$ is a random oracle, $\mathcal{B}_1$ can simulate the role of the CA and answer $\mathcal{A}$'s implicit certificate requests without knowing d by careful pre-selection of the random values of $Hash$. Algorithm $\mathcal{B}_1$ simulates the role of the CA as follows: given a request R_i for an implicit certificate with Q, $\mathcal{B}_1$ generates integers $s_i, h_i \in_R [1, n)$ and computes V_i = $R_i + (s_i * G - h_i * Q)$, $\mathcal{B}_1$ defines $Hash(V_i, t_i, Q)$ = h_i and returns the triple (V_i, t_i, s_i) as the response to $\mathcal{A}$'s request. Since $R_i + s_i * G$ = $V_i + Hash(V_i, t_i, Q) * Q$ holds, the response to the implicit certificate request appears legitimate from $\mathcal{A}$'s perspective. Furthermore, the hash function will be random from $\mathcal{A}$'s perspective because the value h_i was initially chosen randomly.*

The adversary $\mathcal{A}$ is of course allowed to query $Hash$ directly. Given a hash query input, say, (V, t, Q), which has not been previously queried or determined as above, $\mathcal{B}_1$ outputs $H_1(V, t, Q)$ where R is the message on which it is trying to forge a signature, V=$R + k * G$ and k is a random number in $[1, n)$. Clearly, the distribution of the simulated hash values generated by $\mathcal{B}$ will be indistinguishable to $\mathcal{A}$ from the distribution of hash values generated by a random oracle.

Suppose that $\mathcal{A}$ is successful. Then $\mathcal{A}$ returns a tuple (V, t, s) such that $R + s * G$ = $V + h * Q_j$ for some j, with h = $Hash(V, t, Q_j)$, such that either:

(i) (V, t, s) *is an implicit certificate created by CA_j for a request from $user_i$; or*
(ii) (V, t, s) *is an implicit certificate which was not issued by CA_j.*

Assume we are in the first case. Then there is at least a $1/\tau'$ probability that the request R_i of $user_i$ was the challenge point Q given as input to the algorithm $\mathcal{B}_1$. The private key d_i of $user_i$ discovered by $\mathcal{A}$ satisfies d_i = $(r_i + s) \bmod n$. But d = r_i, and $\mathcal{B}_1$ can observe s as CA_k's response. Thus $\mathcal{B}_1$ can compute d = $(d_i - s) mod n$.

Assume we are in the second case. Then there is at least a $1/\tau'$ probability that public key Q_j of CA_j is the challenge point Q given as input to the algorithm $\mathcal{B}_1$. We can assume that (V, t, s) was an input query to the random oracle hash $Hash$, because otherwise the equation $R + s * G = V + h * Q_j$ will hold with negligible probability, contradicting the assumption that ϵ is non-negligible. Thus $Hash(V, t, Q_j)$ = $H_1(V, t, Q_j)$ by definition of the simulation. But now (V, t, s) is a signature of the message R.

There is a minor problem that, if during execution of $\mathcal{B}_1$ with $\mathcal{A}$, the message (V, t, s) appears first as a direct query to $Hash$, and subsequently as an implicit certificate constructed during the simulation of a CA. Since the values k_i and h_i are chosen randomly during simulation of the CA, the point V_i will be uniformly distributed, and thus, this event of V = V_i will happen with negligible probability. Nevertheless, in this case $\mathcal{B}_1$ can simply start over.

Clearly, if $\mathcal{A}$ runs in polynomial time and succeeds with non-negligible probability then so will $\mathcal{B}_1$. By Lemma 1 and above, if $\mathcal{A}$ runs in polynomial time and succeeds with non-negligible probability then so will $\mathcal{B}$. But by hypothesis, it was assumed that no such $\mathcal{B}$ for solving discrete logarithms in $\mathcal{G}$ existed. Therefore no adversary $\mathcal{A}$ exists in the random oracle model unless discrete logarithms in $\mathcal{G}$ can be efficiently solved.

Bibliography

[1] X. Shi, S. Shi, M. W., J. Kaunisto, C. Qian, On-device IoT certificate revocation checking with small memory and low latency. CCS 2021: 1118-1134

[2] J. Viega, M. Messier, P. Chandra, Network security with OpenSSL: Cryptography for secure communications, Sebastopol, CA, USA: O'Reilly Media, 2002.

[3] R. Hummen, J. Hiller, M. Henze, K. Wehrle, Slimfit - A HIP DEX compression layer for the IP-based Internet of Things. WiMob 2013: 259-266.

[4] T. Weil, VPKI hits the highway – secure communication for the US DOT connected vehicle pilot program, May 2017. [Online]. Available: https:// www.securityfeeds.us/sites/default/files/VPKI_Hits_High way_DineLearn_9May2017.pdf

[5] M. M. Hossain, R. Hasan. P-HIP: A lightweight and privacy-aware host identity protocol for Internet of Things. IEEE Internet Things J. 8(1): 555-571 (2021)

[6] P. S. L. M. Barreto, M. A. Simplício Jr., J. E. Ricardini, H. K. Patil, Schnorr-based implicit certification: Improving the security and efficiency of vehicular communications, IEEE Trans. Computers 70(3): 393-399 (2021).

[7] C.G. Günther, An identity-based key-exchange protocol, in: Advances in Cryptology-Eurocrypt'89, 1989, pp. 29-37.

[8] M. Girault, Self-certified public keys, In: Advances in Cryptology-Eurocrypt'91, 1991, pp. 490-497.

[9] D. Brown, R. Gallant, S. Vanstone, Provably secure implicit certificate schemes, in Proc. 5th Int. Conf. Financial Cryptography, 2002, pp. 156–165.

[10] M. Campagna, Sec 4: Elliptic curve qu-vanstone implicit certificate scheme (ECQV), Certicom Res., vol. 4, p. 32, 2013.

[11] P. Nikander, A. Gurtov, T. R. Henderson, Host identity protocol (HIP): connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks, IEEE Commun. Surv. Tutorials 12(2): 186-204 (2010)

[12] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, Transmission of IPV6 packets over IEEE 802.15.4 networks, IETF, RFC 4944, 2007.

[13] K. Nyberg, Fast accumulated hashing, in: Proc. 3rd Int. Workshop Fast Softw. Encryption, 1996, pp. 83–87.

[14] Zolertia, Z1 Mote IoT Device, 2016. [Online]. Available: http://zolertia.sourceforge.net/

[15] B. Brecht, D. Therriault, A.Weimerskirch, W. Whyte, V. Kumar, T. Hehn, R. Goudy, A security credential management system for V2X communications, IEEE Trans. Intell. Transp. Syst. 19(12): 3850-3871 (2018)

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

[17] P. Gope, B. Sikdar, Lightweight and privacy-friendly spatial data aggregation for secure power supply and demand management in smart grids, IEEE Trans. Inf. Forensics Secur. 14(6): 1554-1566 (2019).

[18] PBC library, accessed: Apr. 16, 2017. [Online]. Available: http://crypto.standford.edu/pbc/.

[19] L. Lu, J.Lu, A lightweight verifiable secret sharing in Internet of Things, IJACSA 13(5):1028-1035 (2022).

[20] P.Gope, B.Sikdar, Lightweight and privacy-friendly spatial data aggregation for secure power supply and demand management in smart grids, IEEE Trans. Inf. Forensics Secur. 14(6): 1554-1566 (2019)