Future Generation Computer Systems 113 (2020) 132–144

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Non-interactive certificate update protocol for efficient authentication in IoT☆

Li Duan [a,b,1], Yong Li [b,1], Lijun Liao [b,1,*]

[a] Paderborn University, Germany
[b] Huawei Technologies Duesseldorf GmbH, Germany

ARTICLE INFO

Article history:
Received 31 January 2020
Received in revised form 24 June 2020
Accepted 2 July 2020
Available online 8 July 2020

Keywords:
Authentication
Internet-of-Things
Public key infrastructure
Certificate update
Provable security

ABSTRACT

A non-interactive certificate update protocol (NICU) allows a certification authority (CA) to update the public key of node certificates without interacting with nodes and with the new certificate, a node can update its private signing key accordingly. In this paper, we provide constructions of efficient NICU protocols and instantiate them based on pseudo random functions, encryption schemes and the homomorphism of the signature key pairs. We thoroughly evaluate the NICU protocols and exhibit their outstanding efficiency in terms of speed of certificate update and bandwidth consumption. In addition, we provide a formal security model of NICU, which can be of independent interest, and prove that our protocols are secure.

1. Introduction

No one can deny the trend of mass deployment of Internet-of-Things (IoT) nodes [1]. The huge number of active IoT nodes also means there is a huge amount of data to protect. In practice, as IoT nodes are constrained computation nodes with limited power and resources [2], fake nodes and malicious data injection can not only disrupt the security of the network, but also consume precious energy of other honest nodes [3]. One of the effective ways to mitigate fake nodes is to introduce node level authentication mechanisms into IoT nodes. Without pre-shared keys [4], typical authenticated and confidential channel establishment (ACCE) protocols [5], such as (D)TLS 1.2 [6,7], requires the participants to authenticate each other via certificate checking, as shown in Fig. 1. In principle, if digital certificates and the corresponding services are available, then IoT can rely on them to ensure data security and node authentication.

CA and digital certificates

As a part of a public key infrastructure (PKI), a certification authority (CA), as in Fig. 1, can support authentication by issuing and maintaining digital certificates for different communicating subjects. Digital certificates are data structures which bind public keys to subjects [8]. Given a certificate with public key pk of subject c, when c's ownership of the corresponding private key sk can be confirmed via interaction, subject c can be potentially authenticated.

On the other hand, a certificate has only a limited lifetime and the private key might be compromised by attackers before the certificate expires. Therefore, the CA has to issue a new certificate for subjects to prevent cyber security disasters. This can be done by the CA alone, if a subject c requests the CA to generate both the new public key and the corresponding private key. A more common way is initiated by the certificate owner. A subject c, who wants a new certificate, first generates a public–private key pair (pk, sk) and sends the public key pk to the CA. If c can convince the CA about its identity and ownership of the sk via online interaction, then the CA will issue a new certificate for c [9]. The conventional way to update a certificate is quite similar to the registration of a new certificate.

Conventional certificate provision and update process

The general process to issue and update digital certificates is illustrated in Fig. 2. Before being delivered to customers, IoT nodes are provisioned with initial protection materials, e.g. username/password, key pair, or pre-shared key, in a trusted environment. This environment is also called *production environment*.

Once an IoT device is activated by the customer in the *field environment*, it starts to check regularly whether it still possesses a valid certificate and whether the certificate needs to be updated soon. Once the criteria to enroll or update a certificate is satisfied, the device will (1) generate a new key pair, (2) create a certificate

☆ Extended version of the paper with the same title in ACM IoT 2019.

* Corresponding author.
E-mail address: lijun.liao@huawei.com (L. Liao).

[1] All author have equal contribution and names are listed in alphabetic order.

https://doi.org/10.1016/j.future.2020.07.002
0167-739X/

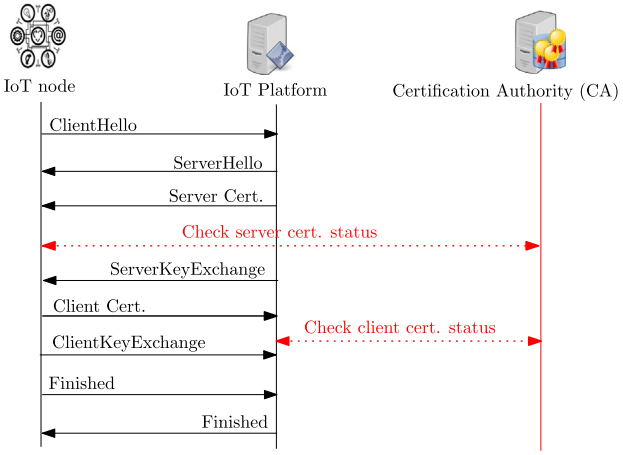

Fig. 1. Authentication in (D)TLS with CA.

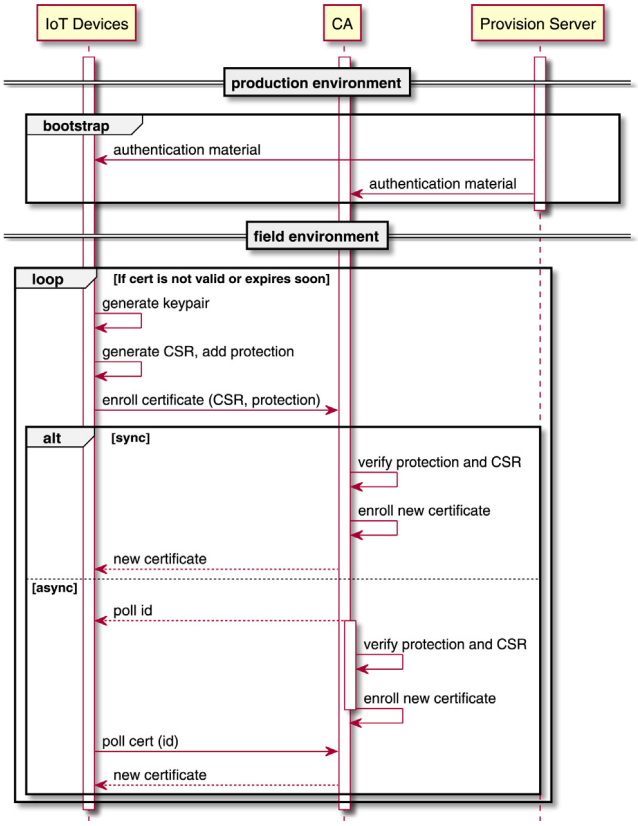

Fig. 2. Conventional certificate provision and update process.

signing request (CSR) containing the new public key signed by the new private key, (3) add protection[2] to the CSR with the pre-provisioned protection material, and finally send the CSR together with the protection to the CA.

In the synchronous mode, the CA (1) verifies the protection, (2) verifies the signature of the CSR, (3) generates a new certificate, and (4) sends the new certificate back to the IoT device. In order to serve a large amount of IoT devices, the CA may switch to the asynchronous mode. In this mode, the CA sends just a poll id in response to the request. The CA can then execute the steps 1 - 3 if it has idle resources. The IoT device polls regularly until it receives the new certificate or a termination error.

The certificate update is usually triggered by certificate registration of new nodes, the revocation of compromised certificates, and updates of old certificates. If certificates are updated short-periodically for security reasons, to verify the rush of certificate requests can also overwhelm CA.

Challenges in IoT node certificate update protocol

For the CA, as shown in Fig. 2, the most costly local operations during the update are generating new key pairs, signing on messages and verifying signatures in CSR. In comparison with a CA for WWW, the huge number of IoT nodes amplifies the computational cost of certificate update. Moreover, to perform certificate update, the CA has to wait for the nodes' new public key, which can prevent the CA from optimal job scheduling.

On the other hand, every message counts for the life of an IoT node. As it is usually only powered by a battery of constrained capacity, reduction of messages means prolonging of the node's lifetime. In addition, it becomes more challenging to increase certificate update frequency supportable by IoT nodes, since it is important to balance the compromise of security (maintaining updated certificate information) and performance (battery power consumption).

Therefore, it is meaningful to design an efficient certificate update protocol to reduce interaction between nodes and CA and local cryptographic operations, while preserving high security.

1.1. Our contribution and the outline of the paper

We use the butterfly key expansion [10] for the optimal use of the initial randomness and design three non-interactive certificate update (NICU) protocols for IoT nodes, aiming at efficiency in practice and provable security.

We also propose formal definitions for the NICU protocol, emphasizing the independence of private keys and certificates if they are in different periods. The formal security analysis and performance evaluation show that our design is robust and extensible.

Notation used in the algorithms and definitions of cryptographic primitives is introduced in Section 3. The abstract syntax and security definitions of NICU are presented in Section 4. The concrete design and theorems for security and performance, as well as comparison with existing butterfly key based solutions, are provided in Section 4. The evaluation of the NICU protocols is presented in Section 5. Other related works are summarized in Section 2. The conclusion and the future work are presented in Section 6.

2. Related works and comparison with existing solutions

Authenticated key exchange protocols and PKI. Diffie and Hellman [11] proposed the first key exchange protocol over an insecure channel in their seminal work. As the most popular secure channel constructing protocols for data confidentiality or authentication, DTLS and TLS also serve as the fundamental security guarantee for upper-layer protocols, such as CoAP [12]. DTLS and TLS are usually used within Public Key Infrastructure (PKI) [8], which uses digital certificates as the principal method for authentication, unless the communicating parties possess pre-shared cryptographic keys [13].

Improving performance of PKI and certificate-based authentication. Although digital certificates have much nicer security features and enable flexibility in credential management, the infrastructure in its original form demands more resources than a

[2] The CSR can be signed with the private key corresponding to the current node certificate or encrypted with the public encryption key of the CA.

typical node can have. Therefore, a PKI has to be adapted before being implemented on constrained devices.

A natural way for the adaptation is to do divide-and-conquer with every component of a PKI, i.e., to replace every heavy building blocks with its lightweight peers. The major components include cryptographic primitives, certificate information maintenance, and key material maintenance. LPKI [14] takes this approach and replaces RSA signature with elliptic curve based signature [15]. The authors also suggest replacing TLS handshake with HMQV. Other researchers employ lightweight PKI components for certificate information maintenance, such as compressed certificate revocation list (CRL) [16] or hierarchical CRL [17]. These efficient CRL mechanisms are also suggested for the browser [18], V2X communication [16] and advanced metering networks [19].

This paper focus on the third component, the key material maintenance. More specifically, we use novel, lightweight, and non-interactive certificate update protocols to replace the conventional and heavy ones.

2.1. Comparison with existing solutions

In 2018, Brecht et al. [10] propose a *butterfly public key* generation method for privacy preserving authentication in Vehicle-to-X communication, which uses a pseudo-random function and encryption schemes to derive new key pairs from the initial keys. To generate butterfly signing keys, the client and server have to be provisioned with the same secret key k and an initial public counter value x.

Although the concrete construction in this paper and in [10] all rely on the homomorphism of ECDSA keys and pseudo-random functions (PRF), the solutions in this paper differ from [10] in the following aspects.

Provably Stronger Security The security guarantee provided by [10] is that the attacker $\mathcal{A}$ cannot fully recover any signing keys, given a set of certificates (public keys) generated so far. However, full recovery of keys might be the weakest form of security that a cryptographic protocol must have. For certificate based authentication schemes, if an attacker $\mathcal{A}$ can forge even one signature with non-negligible probability, the authentication scheme should be considered broken. For example, if $\mathcal{A}$ succeeds in forging a signature on a malicious software patch, then any device that installs this patch will become a potential victim of further attacks. On the other hand, one signature forgery does not necessarily imply the leakage of the whole secret signing key. Therefore, we provide a stronger security definition and design protocols that can meet the requirements. In addition to the set of existing certificates, now $\mathcal{A}$ is given a valid signing key sk and transcripts sent between the communicating parties. Moreover, the NICU scheme is broken if there is some $\mathcal{A}$ who can generate one new signature efficiently. Let pk (cert) be the corresponding public key (certificates) of sk. By new signatures it means that a signature can be verified by any other public key (certificate) $\mathsf{pk}' \neq \mathsf{pk}$ in the set. In principle, we consider a more powerful $\mathcal{A}$ and weaker restrictions. Thus a NICU protocol which can withstand $\mathcal{A}$'s attack has better security assurance.

More modular construction and common building blocks Except for the signature scheme, other components in NICU protocols in this paper can be easily replaced with functionally equivalent building blocks. For example, HMAC [20] and a lot of other password-hashing algorithms can be used as PRF in our constructions.

One-time key instead of long-term key for better security Although not explicitly mentioned in the general threat model, we also consider how the leakage of secret keys can impact the security in our protocol design.

In our NICU1, not only the key pairs for certificates, but also the PRF keys are updated periodically. We integrate reliable PRF key update mechanisms to provide much better key independence. More specifically, the attacker $\mathcal{A}$ cannot recover previous signing keys of NICU1 by compromising the current PRF key and the current signing key. On the other hand, in the solution given by [10], knowing the current key(s)[3] and any signing key enables the attacker to recover all the previous and future keys.

3. Notation and preliminaries

3.1. Notation

We let $\kappa \in \mathbb{N}$ denote the security parameter and 1^κ the string that consists of κ ones. Let $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ be the set of integers between 1 and n. If S is a set, $a \xleftarrow{\$} S$ denotes the action of sampling a uniformly random element from S. If $\mathcal{A}()$ is an algorithm, $m \xleftarrow{\$} \mathcal{A}^{\mathsf{O}(\cdot)}()$ denotes that $\mathcal{A}$ (probabilistically) outputs m with the help of another algorithm $\mathsf{O}(\cdot)$. Let $X \| Y$ denote the operation concatenating two binary strings X and Y. We use $\Pr[A : B]$ to denote the probability that A happens if condition B is met. B can also be some actions taken by attackers. We use $\#E$ to denote the number of objects in category E.

In the following sections, we use genCert() to denote the process of generating a digital certificate by the CA. Other notation is introduced when needed.

3.2. Cryptographic primitives

Definition 1 (*Signature Scheme*)**.** A signature scheme $\mathsf{SIG} = (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Vfy})$ consists of three algorithms SIG.Gen, SIG.Sign and SIG.Vfy described as below.

- $\mathsf{SIG.Gen}(1^\kappa) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$. The non-deterministic key generation algorithm SIG.Gen() takes the security parameter 1^κ as the input and outputs the public key pk and the private key sk.
- $\mathsf{SIG.Sign}(\mathsf{sk}, m) \xrightarrow{\$} \sigma$. The (non-deterministic) message signing algorithm SIG.Sign() takes the private key sk and a message m as the input and outputs the signature σ.
- $\mathsf{SIG.Vfy}(\mathsf{pk}, m, \sigma) = b$. The deterministic signature verification algorithm SIG.Vfy() takes the public key pk, a message m, a signature σ as input and outputs a boolean value b. b is TRUE iff σ is a valid signature on m.

Definition 2 (*Existentially Unforgeable Signature*)**.** Let $\mathsf{O}^{\mathsf{SIG}}_{sk}(\cdot)$ be a signing oracle that outputs a signature σ given message m with the private key sk in accordance to SIG. We say that a signature scheme SIG is existentially unforgeable against chosen message attack if there exists a negligible function $\epsilon_{\mathsf{seuf}}()$ such that for any algorithm $\mathcal{F}$ with access to $\mathsf{O}^{\mathsf{SIG}}_{sk}(\cdot)$ and its running time bounded by $\mathsf{poly}(\kappa)$ it holds that

$$\Pr\left[\begin{array}{c}(m*, \sigma^*) \leftarrow \mathcal{F}^{\mathsf{O}^{\mathsf{SIG}}_{sk}(\cdot)}(1^\kappa) : \\ m* \notin \mathcal{S} \bigwedge \mathsf{SIG.Vfy}(\mathsf{pk}, m, \sigma^*) = \mathsf{TRUE}\end{array}\right] \leq \epsilon_{\mathsf{euf},\mathsf{SIG}}(\kappa),$$

where $\mathcal{S}$ is the set of message–signature pair $\mathcal{F}$ has queried and received from $\mathsf{O}^{\mathsf{SIG}}_{sk}(\cdot)$.

Definition 3 (*Collision-resistant Hash Function*)**.** A hash function $\mathsf{Hash} : \mathcal{M} \to \mathcal{D}$ is collision resistant if there exists a negligible function $\epsilon_{\mathsf{coll},\mathsf{Hash}}()$ such that for any algorithm $\mathcal{A}$ with running time bounded by $\mathsf{poly}(\kappa)$, it holds that

$$\Pr\left[\begin{array}{c}(m_0, m_1) \leftarrow \mathcal{A}(1^\kappa, \mathsf{Hash}) : \\ m_0 \neq m_1 \bigwedge \mathsf{Hash}(m_0) = \mathsf{Hash}(m_1)\end{array}\right] \leq \epsilon_{\mathsf{coll},\mathsf{Hash}}(\kappa),$$

where $\mathcal{M}$ is the message space and $\mathcal{D}$ is the hash image space.

[3] In [10], this includes signing key, PRF key and a decryption key to receive a random value.

Definition 4 (*Pseudo-random Function, PRF*)**.** A pseudo-random function $\mathsf{F} = (\mathsf{FKGen}, \mathsf{PRF})$ consists of two algorithms FKGen and PRF described as below.

- $\mathsf{F.FKGen}(1^\kappa) \xrightarrow{\$} \mathsf{k}$. The non-deterministic key generation algorithm FKGen() takes the security parameter 1^κ as the input and outputs the secret key k.
- $\mathsf{F.PRF}(\mathsf{k}, x) = y$. The PRF evaluation algorithm PRF() takes as the input the secret key k and a value x in the domain and outputs an image y.

Definition 5 (*Public Key Encryption Scheme, PKE*)**.** A public key encryption scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms KGen, Enc and Dec described as below.

- $\Pi.\mathsf{KGen}(1^\kappa) \xrightarrow{\$} (\mathsf{ek}, \mathsf{dk})$. The non-deterministic key generation algorithm KGen() takes the security parameter 1^κ as the input and outputs the public encryption key ek and the private decryption key dk.
- $\Pi.\mathsf{Enc}(\mathsf{ek}, m) \xrightarrow{\$} \mathsf{CT}$. The non-deterministic encryption algorithm Enc() takes the encryption key ek and a message m as the input and outputs a ciphertext CT.
- $\Pi.\mathsf{Dec}(\mathsf{dk}, \mathsf{CT}) = m'$. The deterministic decryption algorithm Dec() takes the public key pk, a ciphertext CT as input and outputs a plaintext m'.

Definition 6 (*Symmetric Key Encryption Scheme, SKE*)**.** A symmetric key encryption scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms KGen, Enc and Dec described as below.

- $\Pi.\mathsf{KGen}(1^\kappa) \xrightarrow{\$} \mathsf{k}$. The non-deterministic key generation algorithm KGen() takes the security parameter 1^κ as the input and outputs one encryption–decryption key k.
- $\Pi.\mathsf{Enc}(\mathsf{k}, m) \xrightarrow{\$} \mathsf{CT}$. The (non-deterministic) encryption algorithm Enc() takes the key k and a message m as the input and outputs a ciphertext CT.
- $\Pi.\mathsf{Dec}(\mathsf{k}, \mathsf{CT}) = m'$. The deterministic decryption algorithm Dec() takes the key k, a ciphertext CT as input and outputs a plaintext m'.

Due to the page limitation, we refer the reader to a cryptography textbook such as [21] for the semantic security definition of all the cryptographic primitives above.

4. NICU protocols

A non-interactive certificate update protocol (NICU) enables the node to derive its new private keys from the previous one and auxiliary information. It also enables the server (CA) to derive the corresponding public key without knowing the new private key.

After a secure initialization, the core of the protocol is how the update materials are generated securely from the current period to the next. As can be seen in the following definition, both sides must be able to generate the identical update material and use different ways to update the private and public key accordingly. Otherwise, the update would be incorrect.

Here we give an abstract syntax of a NICU protocol. To simplify the notation, we omit the subject identifier c in the public–private key pair and in the certificate of node c.

Definition 7 (NICU *Protocol Syntax*)**.** A NICU protocol $\mathsf{U} = (\mathsf{Init}, \mathsf{udGen}, \mathsf{updateSK}, \mathsf{updatePK})$ for a signature scheme SIG is a tuple of the following algorithms.

- $\mathsf{U.Init}(\mathsf{SIG}, 1^\kappa) \xrightarrow{\$} (\mathsf{pk}_0, \mathsf{sk}_0, \mathsf{isv}_0, \mathsf{aux}_0)$. The initialization algorithm Init takes as input (the description of) a signature scheme SIG and the security parameter 1^κ and outputs an updatable public–private key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$ for time $\tau = 0$, an initial secret value isv_0, and auxiliary information aux_0, which can be shared between the node and CA.
- $\mathsf{U.udGen}(\mathsf{isv}_\tau, \mathsf{aux}_{\tau+1}) \to (\mathsf{isv}_{\tau+1}, \mathsf{u}_{\tau+1})$. The update value generation algorithm udGen takes as input the secret value isv_τ and auxiliary data $\mathsf{aux}_{\tau+1}$. It outputs the new secret and update values $(\mathsf{isv}_{\tau+1}, \mathsf{u}_{\tau+1})$
- $\mathsf{U.updatePK}(\mathsf{pk}_\tau, \mathsf{u}_{\tau+1}, \mathsf{aux}_{\tau+1}) = \mathsf{pk}_{\tau+1}$. The public key update algorithm updatePK takes as input the old public key pk_τ, the update value $\mathsf{u}_{\tau+1}$, and auxiliary data $\mathsf{aux}_{\tau+1}$. It outputs the new public key $\mathsf{pk}_{\tau+1}$.
- $\mathsf{U.updateSK}(\mathsf{sk}_\tau, \mathsf{u}_{\tau+1}, \mathsf{aux}_{\tau+1}) = \mathsf{sk}_{\tau+1}$. The private key update algorithm updateSK takes as input the old private key sk_τ, the update value $\mathsf{u}_{\tau+1}$, and auxiliary data $\mathsf{aux}_{\tau+1}$. It outputs the new secret key $\mathsf{sk}_{\tau+1}$.

Note that aux_τ is additional information needed for computing the new key. It can be $\emptyset$, remain identical all the time, or be updated from period to period. The value depends on the concrete design of the protocol.

Definition 8 (*Correctness of* NICU)**.** A NICU protocol U is correct for a signature scheme SIG given security parameter 1^κ, if for all time point $\tau \geq 0$ and for all message m supported by SIG, it holds that

$$\begin{aligned}\Pr[\mathsf{SIG.Vfy}(\mathsf{pk}_\tau, m, \sigma) = 1 :\ & \\ & \sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_\tau, m), \\ & (\mathsf{pk}_0, \mathsf{sk}_0, \mathsf{isv}_0) \leftarrow \mathsf{U.Init}(\mathsf{SIG}, 1^\kappa), \\ & (\mathsf{isv}_\tau, \mathsf{u}_\tau) \leftarrow \mathsf{U.udGen}(\mathsf{isv}_{\tau-1}, \mathsf{aux}_\tau), \\ & \mathsf{pk}_\tau \leftarrow \mathsf{U.updatePK}(\mathsf{pk}_{\tau-1}, \mathsf{u}_\tau, \mathsf{aux}_\tau) \textbf{ if } \tau \geq 1, \\ & \mathsf{sk}_\tau \leftarrow \mathsf{U.updateSK}(\mathsf{sk}_{\tau-1}, \mathsf{u}_\tau, \mathsf{aux}_\tau) \textbf{ if } \tau \geq 1] = 1\end{aligned}$$

Rationale of NICU *syntax*

1. A NICU protocol is always bound with a (class of) signature scheme SIG. A NICU for RSA signature might not function with ECDSA.
2. The public key and private keys should always be conforming with SIG, if the NICU is correct. More specifically, not only the length and form of each key at each time point should be conforming with SIG for given security parameter, but each signature σ signed with private key sk_τ should pass the verification using pk_τ.

A digital certificate cert_i contains the corresponding public key pk_i, a signature by CA over pk_i, and other auxiliary information. We use $\mathsf{pk}_i \in \mathsf{cert}_i$ to denote that pk_i is contained in cert_i and cert_i is valid.

Definition 9 (*Forward Certificate Independence*)**.** A NICU protocol has single node forward certificate independence, if for all time point $\tau \geq 0$, all (probabilistic) algorithm $\mathcal{A}$ with running time polynomially in κ, and all $0 < \gamma < \tau$, the advantage of $\mathcal{A}$ is negligible. More specifically,

$$\begin{aligned}\mathsf{Adv}_{\mathsf{NICU,FW}} = \Pr\big[\mathsf{SIG.Vfy}(\mathsf{pk}_{\tau-\gamma}, m, \sigma) = 1 :\ & \\ & \mathcal{A}\left(\mathsf{sk}_\tau, \mathsf{T}_\tau, \{\mathsf{cert}_\alpha\}_{\alpha \leq \tau}\right) \to (m, \sigma, \gamma), \\ & \mathsf{pk}_{\tau-\gamma} \in \mathsf{cert}_{\tau-\gamma} \bigwedge m \notin \mathsf{T}_\tau\big] \leq \mathit{negl}(\kappa)\end{aligned}$$

where T_τ is the transcript, including all messages sent, received or signed by the node, until time τ, $\mathit{negl}(\kappa)$ a negligible function of κ, and the probability is taken over all choice of sk_τ and the random coins used by $\mathcal{A}$.

Definition 10 (*Backward Key Independence*)**.** Let τ_{max} be the maximal number of certificate updates allowed, which is polynomial

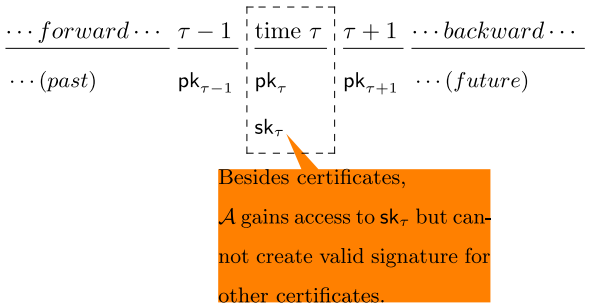

Fig. 3. Forward and backward certificate independence.

in the security parameter κ. A NICU protocol has single node forward certificate independence, if for all time point τ, all (probabilistic) polynomial time algorithm $\mathcal{A}$, and all $\gamma > 0$, the advantage of $\mathcal{A}$ is negligible. More specifically,

$$\mathsf{Adv}_{\mathsf{NICU,BW}} = \Pr\Big[\mathsf{SIG.Vfy}(\mathsf{pk}_{\tau+\gamma}, m, \sigma) = 1 : \mathcal{A}\left(\mathsf{sk}_{\tau}, \mathsf{T}_{\tau_{max}}, \{\mathsf{cert}_{\alpha}\}_{\alpha \le \tau_{max}}\right) \rightarrow (m, \sigma, \gamma), \mathsf{pk}_{\tau+\gamma} \in \mathsf{cert}_{\tau+\gamma} \bigwedge m \notin \mathsf{T}_{\tau_{max}} \bigwedge (\tau + \gamma) \le \tau_{max}\Big] \le negl(\kappa)$$

where $\mathsf{T}_{\tau_{max}}$ is the transcript, including all messages sent, received or signed by the node, until time τ_{max}, $negl(\kappa)$ a negligible function of κ, and the probability is taken over all choice of sk_{τ} and the random coins used by $\mathcal{A}$.

Definition 11 (*Strictly Non-interactive* NICU)**.** A NICU is strictly non-interactive if for all $\tau \ge 0$

$$\Pr[\mathsf{U.udGen}\left(\mathsf{isv}_{\tau}, \emptyset \cup \{\mathsf{cert}_{\tau}\}\right) = \mathsf{U.udGen}\left(\mathsf{isv}_{\tau}, \mathsf{T}_{\tau} \cup \{\mathsf{cert}_{\tau}\}\right)] = 1$$

Rationale of security definition

1. With the security definition, we try to capture the requirement that a node certificate, once updated, has no exploitable connection with any private key other than its updated one.
   A NICU has *forward* certificate independence, if a new signing key cannot help an adversary $\mathcal{A}$ forge signatures verifiable with *past* certificates. More specifically, even if $\mathcal{A}$ is given all the current available information, including current signing key, all used certificates till now, and the possible transcript of authentication, it remains hard for $\mathcal{A}$ to forge even one signature which can pass the verification using any old certificate. This also implies the impossibility of recovering any old private signing key.
   The definition of *backward* certificate independence is similar, i.e., it is hard to generate *future* signatures given the current information. The relationship of time points is illustrated in Fig. 3.
2. A NICU is strictly non-interactive, if whether the transcript exists is irrelevant for computing key update value u.

Our NICU protocols rely on the properties of Elliptic Curve Digital Signature Algorithm (ECDSA) [22], so we give a brief introduction of it here.

Definition 12 (*ECDSA Signature Scheme*)**.** Let the public parameter of group of elliptic curve points over a finite field be $(\mathbb{G}, \mathcal{P}, q)$, where $\mathbb{G}$ is the description of the group, $\mathcal{P}$ a base point (a generator), and q the order of $\mathcal{P}$. Let $h()$ be a cryptographic hash function which maps bit-string to an integer in $\{1, \ldots, q-1\}$. The ECDSA signature scheme SIG is defined as follows.

- $\mathsf{SIG.Gen}(1^{\kappa})$: Pick $\mathsf{sk} \xleftarrow{\$} \{1, \ldots, q-1\}$ and set $\mathsf{pk} = \mathsf{sk} \cdot \mathcal{P}$. Output $(\mathsf{pk}, \mathsf{sk})$
- $\mathsf{SIG.Sign}(\mathsf{sk}, m)$: Select $k \xleftarrow{\$} \{1, \ldots, q-1\}$ and compute $(x_1, y_1) = k \cdot \mathcal{P}$. Set $r = x_1 \mod q$, then compute $s = k^{-1}(h(m) + \mathsf{sk} \times r) \mod q$. Output (r, s) as the signature σ.
- $\mathsf{SIG.Vfy}(\mathsf{pk}, m, \sigma)$:
  - Parse σ as (r, s) and verify that both r and s is in $\{1, \ldots, q-1\}$. Compute $u_1 = h(m) \times s^{-1} \mod q$ and $u_2 = r \times s^{-1} \mod q$.
  - Compute $(x_0, y_0) = u_1 \cdot \mathcal{P} + u_2 \cdot \mathcal{P}$ and set $v = x_0$. Output 1 iff $v = r$

Homomorphism of ECDSA keys. Given the public group parameter $(\mathbb{G}, \mathcal{P}, q)$, key pair $(\mathsf{pk}, \mathsf{sk})$, and an integer u, it can be seen that $(\mathsf{pk} + (\mathsf{u} \mod q) \cdot \mathcal{P}, \mathsf{sk} + \mathsf{u} \mod q)$ is also a valid ECDSA key pair for the same EC group. Moreover, given u, the computation of $\mathsf{pk} + (\mathsf{u} \mod q) \cdot \mathcal{P}$ can be done without knowing sk or $\mathsf{sk} + \mathsf{u} \mod \mathcal{P}$. This feature provides foundation for updating public key and generating the corresponding certificate in a non-interactive way.

The following three NICU protocols assume different capacity of IoT nodes and CA. NICU1 needs the node to support PRF algorithm and secure storage for an additional PRF key. NICU2 needs public encryption scheme and secure storage for the key pair, while NICU3 needs symmetric encryption scheme and also storage for the symmetric encryption key.

4.1. *PRF-based* NICU1

We first propose an extremely efficient NICU protocol NICU1 as shown in Fig. 4, which is strictly non-interactive and with forward secrecy of device signing keys. Besides those required by X.509 [8] certificate, the only cryptographic primitive used in NICU1 is pseudo-random function (PRF) for key update material derivation. Compared with the solution [10], which use pre-shared and fixed PRF keys, our public key generation method uses updatable PRF keys. In the description of following protocols, we will use the term device and node interchangeably.

The NICU1 functions as follows.

- **Initialization**. $\mathsf{U.Init}(\mathsf{SIG}, 1^{\kappa}) \xrightarrow{\$} (\mathsf{pk}_0, \mathsf{sk}_0, \mathsf{isv}_0, \mathsf{aux}_0)$: The device manufacturer generates the device initial public key $(\mathsf{pk}_0, \mathsf{sk}_0)$, chooses a random PRF key ku_0, and sets up these keys in the device as $\mathsf{isv}_0 := \mathsf{ku}_0$. Then the public key pk_0 and ku_0 are delivered to CA through secure channel. aux_0 is set to be $\emptyset$.
- **Certificate update at time** $\tau + 1$. The device and the server can carry out the update independently, given the certificate cert_{τ} and the PRF key ku_{τ} used previously at τ.
  - $\mathsf{U.udGen}(\mathsf{isv}_{\tau}, \mathsf{aux}_{\tau+1}) \rightarrow (\mathsf{isv}_{\tau+1}, \mathsf{u}_{\tau+1})$:
    Let $\mathsf{isv}_{\tau} := \mathsf{ku}_{\tau}$ for all τ. The new PRF key $\mathsf{ku}_{\tau+1}$ and the $\mathsf{u}_{\tau+1}$ are derived from the old PRF key ku_{τ} via calling $\mathsf{PRF}(\mathsf{ku}_{\tau}, \mathsf{cert}_{\tau})$.
  - **At CA**. $\mathsf{U.updatePK}(\mathsf{pk}_{\tau}, \mathsf{u}_{\tau+1}, \mathsf{aux}_{\tau+1}) = \mathsf{pk}_{\tau+1}$:
    Here, $\mathsf{aux}_{\tau+1}$ can be $\emptyset$. The new public key $\mathsf{pk}_{\tau+1}$ is obtained by "adding" $\mathsf{u}_{\tau} \cdot \mathcal{P}$ to pk_{τ} and the new certificate is generated by the server in the conventional way.
  - **At device**. $\mathsf{U.updateSK}(\mathsf{sk}_{\tau}, \mathsf{u}_{\tau+1}, \mathsf{aux}_{\tau+1}) = \mathsf{sk}_{\tau+1}$:
    Let $\mathsf{sk}'_{\tau+1}$ be the "sum" of $\mathsf{u}_{\tau+1}$ and sk_{τ}. Let $\mathsf{pk}'_{\tau+1}$ the "sum" of $\mathsf{u}_{\tau} \cdot \mathcal{P}$ and pk_{τ}.
    The data $\mathsf{aux}_{\tau+1}$ is the new certificate $\mathsf{cert}_{c,\tau+1}$, with $\mathsf{pk}_{\tau+1} \in \mathsf{cert}_{c,\tau+1}$. The node set his new private key to be $\mathsf{sk}_{\tau+1} := \mathsf{sk}'_{\tau+1}$ only if $\mathsf{cert}_{c,\tau+1}$ is valid and $\mathsf{pk}'_{\tau+1} = \mathsf{pk}_{\tau+1}$.

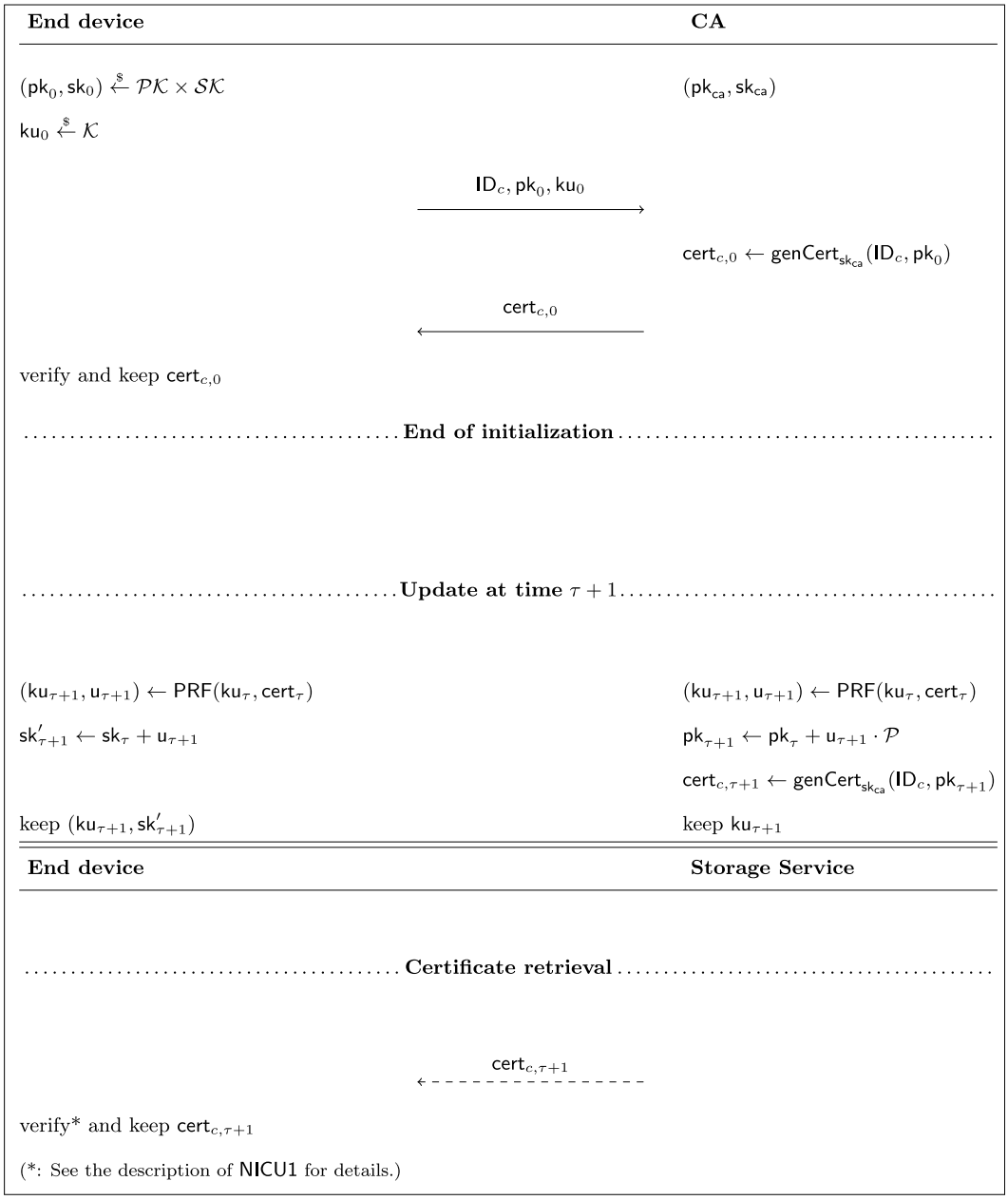

Fig. 4. Non-interactive certificate update protocol NICU1 based on PRF. Certificate retrieval is included for completeness.

A formal security game played between an adversary (attacker) and a challenger is frequently used to model the threat faced by cryptographic protocols. The challenger has a secret in which the adversary is interested, and the adversary can interact with the challenger via different queries. The queries are like application programming interface (API) of a protocol or other possible ways of leaking information. At the end of each game, the adversary outputs what it has newly learned about the secret. For example, the attacker $\mathcal{A}$ against a digital signature scheme can query a signing algorithm $\mathcal{O}()$ to see various message–signature pairs and $\mathcal{A}$ will output a signature σ on a new message m at the end of the game. $\mathcal{A}$ wins if σ is valid and m has not been signed by $\mathcal{O}()$ before, as in Definition 2.

By linking the winning probability of $\mathcal{A}$ in a game for protocol Π with the probability of solving computationally hard problems or breaking existing cryptographic algorithms through reduction, the security of Π can be quantitatively inferred. For complicated protocols or schemes, Shoup proposed the sequence of game technique [23] to organize the reduction between different games and problems and we use it to prove the following theorems.

Theorem 1 (*Security of* NICU1)**.** NICU1 *has single node forward key independence, if the given hash function* Hash *is collision-resistant, the given signature scheme* SIG *is existentially unforgeable, and the function* PRF *is a secure pseudo-random function. More specifically,*

$$\mathsf{Adv}_{\mathsf{NICU1,FW}} \leq \epsilon_{\mathsf{coll,Hash}} + \tau \cdot \left(\epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{euf,SIG}}\right), \tag{1}$$

where $\epsilon_{\mathsf{coll,Hash}}$ *is the probability to find collision of hash function* Hash(), ϵ_{PRF} *the advantage of any probabilistic polynomial time (PPT) algorithm against* PRF, *and* $\epsilon_{\mathsf{euf,SIG}}$ *the advantages of any PPT algorithm against* SIG.

The generation, transmission and storage of all initial cryptographic material is assumed to be completed in a trusted production environment. Any leakage of these secrets can lead to trivial breaks of the NICU protocols presented in this paper.

We further assume the protection of the CA is perfect in the following security proofs, since a compromised CA can simply issue rogue certificates for any given identity, breaking the corresponding public key infrastructure in a trivial way. In practice, people usually use a hash function to compress a long message into a digest of fixed length. The signature is then computed over this digest. Thus we also take Hash into consideration when analyzing protocol security.

Proof. Let $\mathcal{A}$ be an adversary against NICU1, which can break the forward key independence. Let Adv_n be the advantage of $\mathcal{A}$ in security game Game n. We assume the number of signed messages is polynomial in the security parameter κ and we omit κ in the following proof.

Game 0. This is the original security game in Definition 9, where $\mathcal{A}$ gets all the certificates till time τ (inclusive), all the message–signature pairs $\{m_i, \sigma_i\}$ signed with any $\mathsf{sk}_{\tau-\gamma}$, $0 \le \gamma \le \tau$. So we have

$$\mathsf{Adv}_{\mathsf{NICU1,FW}} = \mathsf{Adv}_0 \tag{2}$$

Game 1. We add the following abort rule in this game. If there exist m_i, m_j such that $m_i \neq m_j$ and $\mathsf{Hash}(m_i) = \mathsf{Hash}(m_j)$, abort the game. Note that now Game 1 aborts exactly when a collision of the hash function is found. With a simple probability argument we have

$$\mathsf{Adv}_0 \le \mathsf{Adv}_1 + \epsilon_{\mathsf{coll,Hash}} \tag{3}$$

Game 2. We add the following abort rule in this game. We let the challenger guess the value γ in advance, where γ is the output by $\mathcal{A}$ at the end of this game. If the guess is wrong, abort the game. As the probability of not aborting is at least $\frac{1}{\tau}$, we have

$$\mathsf{Adv}_1 \le \tau \cdot \mathsf{Adv}_2 \tag{4}$$

Game 3. We replace the pseudo-random function with a real random function in this game. If the distribution of $\mathcal{A}$'s output has a non-negligible change in this game compared to in Game 2, we can construct a distinguisher against PRF from $\mathcal{A}$ with advantage bounded by $|\mathsf{Adv}_2 - \mathsf{Adv}_3|$. Therefore we have

$$\mathsf{Adv}_2 \le \mathsf{Adv}_3 + \epsilon_{\mathsf{PRF}} \tag{5}$$

Please note that after Game 3, $\mathsf{sk}_{\tau-\gamma}$ becomes information theoretically independent of any other keys generated before or after $\tau - \gamma$.

Game 4. In this game, we let the challenger randomly generate valid key pairs $(\mathsf{pk}_t, \mathsf{sk}_t), t \le \tau$ conforming with the public parameters. This modification does not change the distribution of keys as in Game 3, because the update values can be seen as the output of an implicitly defined random function. So first we have

$$\mathsf{Adv}_3 = \mathsf{Adv}_4 \tag{6}$$

Now we show how an attacker $\mathcal{F}$ against SIG could be constructed from $\mathcal{A}$. $\mathcal{F}$ is given the public parameters PP, the access to a signing oracle $\mathsf{O}^{\mathsf{SIG}}_{sk*}(\cdot)$ of a SIG instance, and the corresponding public key $\mathsf{pk}*$, as in Definition 2.

To play Game 4 with $\mathcal{A}$ and use $\mathcal{A}$'s output, $\mathcal{F}$ takes the role of the challenger and declares the public parameter of the security game. Except for time $\tau - \gamma$, $\mathcal{F}$ generates valid key pairs $\{(\mathsf{pk}_t, \mathsf{sk}_t)\}, t \le \tau, t \neq \tau - \gamma$ conforming with the public parameters and give $(\mathsf{pk}_\tau, \mathsf{sk}_\tau)$ to $\mathcal{A}$. The distribution of keys remains the same, though $\mathcal{F}$ does not know $\mathsf{sk}*$.

Note that $\mathcal{F}$ can sign all the necessary messages, either by using the signing keys generated by itself or by querying $\mathsf{O}^{\mathsf{SIG}}_{sk*}(\cdot)$. Thus $\mathcal{F}$ can simulate Game 4 perfectly.

If $\mathcal{A}$ wins Game 4, then the game does not abort and $\mathcal{A}$ outputs some valid (m, σ). So our guess of $\tau - \gamma$ is correct, $\mathsf{SIG.Vfy}(\mathsf{pk}_{\tau-\gamma}, m, \sigma) = 1$, and m must be a new message. As $\mathsf{pk}_{\tau-\gamma} = \mathsf{pk}*$, $\mathcal{F}$ can output (m, σ) as a successful forgery w.r.t $\mathsf{pk}*$. Because the probability of forging signatures for SIG is bounded by $\epsilon_{\mathsf{euf,SIG}}$, so is the probability for $\mathcal{A}$ to win Game 4. Therefore, we finally have

$$\mathsf{Adv}_4 \le \epsilon_{\mathsf{euf,SIG}} \tag{7}$$

By combining (2) to (7), Theorem 1 is proved.

4.2. Public key encryption based NICU2

We propose another NICU protocol, which is both forward and backward secure but not strictly non-interactive. Instead of PRF, we use a secure public key encryption scheme, the decryption key of which is protected by the device with special purpose hardware as a long term secret (see Fig. 5).

The NICU2 functions as follows.

- **Initialization**. $\mathsf{U.Init}(\mathsf{SIG}, 1^\kappa) \xrightarrow{\$} (\mathsf{pk}_0, \mathsf{sk}_0, \mathsf{isv}_0, \mathsf{aux}_0)$:
  The device manufacturer generates the node initial public–private key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$ of signature scheme SIG for time $\tau = 0$, chooses a random encryption–decryption key pair $(\mathsf{ek}_c, \mathsf{dk}_c)$, and sets up these keys in the device. Then the public keys pk_0 and $\mathsf{aux}_0 := \mathsf{ek}_c$ are delivered to CA through a secure channel.
- **Certificate update at time** $\tau + 1$. The device and the server can carry out the update almost independently, given the certificate cert_τ and the encryption key ek_c from aux_0.
  - **At CA**. $\mathsf{U.udGen}(\mathsf{isv}_\tau, \mathsf{aux}_{\tau+1}) \to (\mathsf{isv}_{\tau+1}, \mathsf{u}_{\tau+1})$:
    The new update material $\mathsf{u}_{\tau+1}$ is chosen by the CA uniformly at random, i.e., independent of isv and aux.
    $\mathsf{U.updatePK}(\mathsf{pk}_\tau, \mathsf{u}_{\tau+1}, \mathsf{aux}_{\tau+1}) = \mathsf{pk}_{\tau+1}$:
    The new public key $\mathsf{pk}_{\tau+1}$ is obtained by "adding" $\mathsf{u}_\tau \cdot \mathcal{P}$ to pk_τ and the new certificate is generated by the server in the conventional way. In addition, the CA also generates a ciphertext $\mathsf{CT}_{c,\tau+1}$ by encrypting $\mathsf{u}_{\tau+1}$ with the encryption key ek_c of device c, where ek_c can be retrieved from aux_0. For clarity, we write $\mathsf{CT}_{c,\tau+1}$ separately from the certificate but $\mathsf{CT}_{c,\tau+1}$ can be set as an extension in the certificate.
  - **At device**. $\mathsf{U.updateSK}(\mathsf{sk}_\tau, \mathsf{u}_{\tau+1}, \mathsf{aux}_{\tau+1}) = \mathsf{sk}_{\tau+1}$:
    By decrypting $\mathsf{CT}_{c,\tau+1}$ with dk_c, the device obtains $\mathsf{u}_{\tau+1}$.
    Let $\mathsf{sk}'_{\tau+1}$ be the "sum" of $\mathsf{u}_{\tau+1}$ and sk_τ. Let $\mathsf{pk}'_{\tau+1}$ the "sum" of $\mathsf{u}_\tau \cdot \mathcal{P}$ and pk_τ.
    The data $\mathsf{aux}_{\tau+1}$ is the new certificate $\mathsf{cert}_{c,\tau+1}$, with $\mathsf{pk}_{\tau+1} \in \mathsf{cert}_{c,\tau+1}$. The node set his new private key to be $\mathsf{sk}_{\tau+1} := \mathsf{sk}'_{\tau+1}$ only if $\mathsf{cert}_{c,\tau+1}$ is valid and $\mathsf{pk}'_{\tau+1} = \mathsf{pk}_{\tau+1}$.

Theorem 2 (*Security of* NICU2)**.** NICU2 *has both forward and backward key independence, if the given hash function* Hash *is collision-resistant, the signature scheme* SIG *is existentially unforgeable, and the encryption scheme* Π *is secure against chosen ciphertext attacks (CCA) [21]. More specifically,*

$$\mathsf{Adv}_{\mathsf{NICU2,FW}} \le \epsilon_{\mathsf{coll,Hash}} + \tau \cdot (\epsilon_{\mathsf{euf,SIG}} + 2\epsilon^{cca}_{\Pi}) \tag{8}$$

$$\mathsf{Adv}_{\mathsf{NICU2,BW}} \le \epsilon_{\mathsf{coll,Hash}} + (\tau_{max} - \tau) \cdot (\epsilon_{\mathsf{euf,SIG}} + 2\epsilon^{cca}_{\Pi}) \tag{9}$$

where $\epsilon_{\mathsf{coll,Hash}}$ *is the probability for any probabilistic polynomial time (PPT) algorithm to find collision of hash function* Hash(), ϵ_{Π} *the advantage of any PPT algorithm against* Π, *and* $\epsilon_{euf,\mathsf{SIG}}$ *the advantages of any PPT algorithm against* SIG.

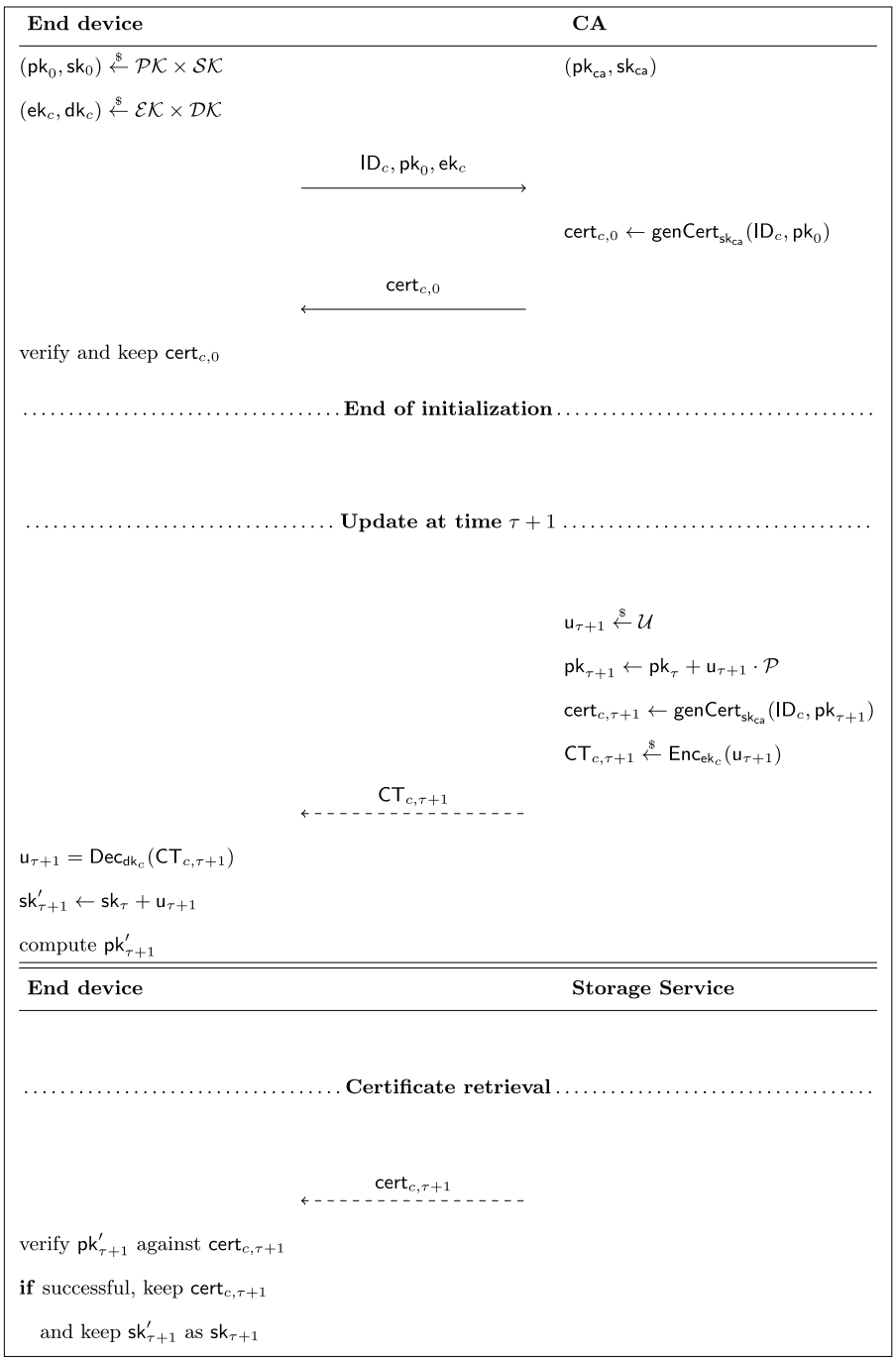

Fig. 5. Non-interactive certificate update protocol NICU2 based on public key encryption. Certificate retrieval is included for completeness.

Proof. We still assume a trusted setup and perfectly secure CA. Since the proof of (8) and (9) are very similar, we prove only (8).

Let $\mathcal{A}$ be an adversary against NICU2, which can break the forward key independence. Let Adv_n be the advantage of $\mathcal{A}$ in security game Game n. We assume the number of signed messages is polynomial in the security parameter κ and we omit κ in the following proof.

Game 0. This is the original security game in Definition 9, where $\mathcal{A}$ gets all the certificates until time τ (inclusive), all the message–signature pairs $\{m_i, \sigma_i\}$ signed with any $\mathsf{sk}_{\tau-\gamma}$, $0 \le \gamma \le \tau$. So we have

$$\mathsf{Adv}_{\mathsf{NICU2,FW}} = \mathsf{Adv}_0 \tag{10}$$

Game 1. We add the following abort rule in this game. If there exist m_i, m_j such that $m_i \neq m_j$ and $\mathsf{Hash}(m_i) = \mathsf{Hash}(m_j)$, abort the game. Note that now Game 1 aborts exactly when a collision of the hash function is found. With a simple probability argument we have

$$\mathsf{Adv}_0 \le \mathsf{Adv}_1 + \epsilon_{\mathsf{coll,Hash}} \tag{11}$$

Game 2. We add the following abort rule in this game. We let the challenger guess the value γ in advance, where γ is the output by $\mathcal{A}$ at the end of this game. If the guess is wrong, abort the game. As the probability of not aborting is at least $\frac{1}{\tau}$, we have

$$\mathsf{Adv}_1 \le \tau \cdot \mathsf{Adv}_2 \tag{12}$$

Game 3. We replace $\mathsf{CT}_{\tau-\gamma}, \mathsf{CT}_{\tau-\gamma+1}$ with encryption of two random values not equal to $\mathsf{sk}_\tau - \mathsf{sk}_{\tau-1}$ nor $\mathsf{sk}_{\tau+1} - \mathsf{sk}_\tau$. If the distribution of $\mathcal{A}$'s output has a non-negligible change in this game compared to in Game 2, we can construct an attacker against Π from $\mathcal{A}$ with advantage $\mathsf{Adv}_2 - \mathsf{Adv}_3$. Since this advantage is also bounded by $2\epsilon_\Pi$, we have

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + 2\epsilon_\Pi^{cca} \tag{13}$$

The modification in the game above makes $\mathsf{sk}_{\tau-\gamma}$ information theoretically independent of the update value $\mathsf{u}_{\tau-\gamma}$ and other keys.

Game 4. This game is unmodified from Game 3, So first we have

$$\mathsf{Adv}_3 = \mathsf{Adv}_4 \tag{14}$$

Now we show how an attacker $\mathcal{F}$ against SIG could be constructed from $\mathcal{A}$. $\mathcal{F}$ is given the public parameters PP, the access to a signing oracle $\mathsf{O}^{\mathsf{SIG}}_{sk*}(\cdot)$ of a SIG instance, and the corresponding public key $\mathsf{pk}*$, as in Definition 2.

To play Game 4 with $\mathcal{A}$ and use $\mathcal{A}$'s output, $\mathcal{F}$ takes the role of the challenger and broadcast the public parameter, the public key of the ca in the security game. $\mathcal{F}$ also generates the encryption–decryption key pair $\mathsf{ek}_c, \mathsf{dk}_c$ and tells $\mathcal{A}$ the public encryption key ek_c. Except for time $\tau - \gamma$, $\mathcal{F}$ generates valid key pairs $\{(\mathsf{pk}_t, \mathsf{sk}_t)\}, t \leq \tau, t \neq \tau - \gamma$ conforming with the public parameters and give $(\mathsf{pk}_\tau, \mathsf{sk}_\tau)$ to $\mathcal{A}$. The distribution of keys remains the same, though $\mathcal{F}$ does not know $\mathsf{sk}*$.

Note that $\mathcal{F}$ can sign all the necessary messages, either by using the signing keys generated by itself or by querying $\mathsf{O}^{\mathsf{SIG}}_{sk*}(\cdot)$. Thus $\mathcal{F}$ can simulate Game 4 perfectly.

If $\mathcal{A}$ wins Game 4, then the game does not abort and $\mathcal{A}$ outputs some valid (m, σ). So our guess of $\tau - \gamma$ is correct, $\mathsf{SIG.Vfy}(\mathsf{pk}_{\tau-\gamma}, m, \sigma) = 1$, and m must be a new message. As $\mathsf{pk}_{\tau-\gamma} = \mathsf{pk}*$, $\mathcal{F}$ can output (m, σ) as a successful forgery w.r.t $\mathsf{pk}*$. Because the probability of forging signatures for SIG is bounded by euf, SIG, so is the probability for $\mathcal{A}$ to win Game 4. Therefore, we finally have

$$\mathsf{Adv}_4 \leq \epsilon_{\mathsf{euf,SIG}} \tag{15}$$

By combining (10) to (15), the bound (8) in Theorem 2 is proved. To prove (9), the guess of γ is in the interval $[1, \tau_{max} - \tau]$. The other necessary games are almost identical to those above.

Remark. The encryption and the late public key verification using the new certificate actually form an authenticated encryption, which provides both confidentiality and authenticity of the update material and prevents the following attack. Suppose $\mathcal{A}$ can control the network. $\mathcal{A}$ can first send to the device $CT_\tau = \mathsf{Enc}_{\mathsf{ek}_c}(\mathsf{u}*)$ at time τ with a $\mathsf{u}*$ known to him, then ask for sk_τ after the updates. By computing $\mathsf{sk}_{\tau-1} = \mathsf{sk}_\tau - \mathsf{u}*$ and a signature on a new m with $\mathsf{sk}_{\tau-1}$, it seems that $\mathcal{A}$ can break the forward key independence. However, since the device can always verify its public key pk_τ against the certificate, which is generated by a secure CA and thus $\mathcal{A}$ cannot forge, the device can always detect and reject the wrong key and update materials. Hence this attack will not succeed with non-negligible probability.

4.3. Symmetric encryption based NICU3

We propose another NICU protocol, which is both forward and backward secure but not strictly non-interactive. Instead of PRF, we use a secure symmetric encryption scheme, the key of which is protected by the device with special purpose hardware as a long term secret (see Fig. 6).

The NICU3 functions as follows.

- **Initialization**. $\mathsf{U.Init}(\mathsf{SIG}, 1^\kappa) \xrightarrow{\$} (\mathsf{pk}_0, \mathsf{sk}_0, \mathsf{isv}_0, \mathsf{aux}_0)$:
  The device manufacturer generates the device initial public–private key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$ of signature scheme SIG for time $\tau = 0$. Then the public key pk_0 is delivered to CA through secure channel. CA generate a certificate $\mathsf{cert}_{c,0}$, derive a node specific symmetric key k_c by using PRF with a long-term secret key $\mathsf{mk}_{\mathsf{ca}}$ and identity c as input, and send back to device $(\mathsf{cert}_{c,0}, \mathsf{k}_c)$. At the node, isv_0 is set to be k_c and aux_0 is $\emptyset$.
- **Certificate update at time** $\tau + 1$. The device and the server can carry out the update almost independently, given the certificate cert_τ and the symmetric key k_c.
  - **At CA**. $\mathsf{U.udGen}(\mathsf{isv}_\tau, \mathsf{aux}_{\tau+1}) \to (\mathsf{isv}_{\tau+1}, \mathsf{u}_{\tau+1})$:
    The new update material $\mathsf{u}_{\tau+1}$ is chosen by the CA uniformly at random, i.e., independent of isv and aux.
    $\mathsf{U.updatePK}(\mathsf{pk}_\tau, \mathsf{u}_{\tau+1}, \mathsf{aux}_{\tau+1}) = \mathsf{pk}_{\tau+1}$:
    The new public key $\mathsf{pk}_{\tau+1}$ is obtained by "adding" $\mathsf{u}_\tau \cdot \mathcal{P}$ to pk_τ and the new certificate is generated by the server in the conventional way. In addition, the CA also generates a ciphertext $\mathsf{CT}_{c,\tau+1}$ by encrypting $\mathsf{u}_{\tau+1} \| (\tau + 1)$ with the encryption key k_c derived from the identity c, and sends $\mathsf{CT}_{c,\tau+1}$ to the device.[4]
  - **At device**. $\mathsf{U.updateSK}(\mathsf{sk}_\tau, \mathsf{u}_{\tau+1}, \mathsf{aux}_{\tau+1}) = \mathsf{sk}_{\tau+1}$
    By successfully decrypting $\mathsf{CT}_{c,\tau+1}$ with k_c, the device obtains $\mathsf{u}_{\tau+1} \| \tau'$.
    If $\tau' = \tau + 1$, continue to the next step. Otherwise abort. Let $\mathsf{sk}'_{\tau+1}$ be the "sum" of $\mathsf{u}_{\tau+1}$ and sk_τ. Let $\mathsf{pk}'_{\tau+1}$ the "sum" of $\mathsf{u}_\tau \cdot \mathcal{P}$ and pk_τ.
    The data $\mathsf{aux}_{\tau+1}$ is the new certificate $\mathsf{cert}_{c,\tau+1}$, with $\mathsf{pk}_{\tau+1} \in \mathsf{cert}_{c,\tau+1}$. The node set his new private key to be $\mathsf{sk}_{\tau+1} := \mathsf{sk}'_{\tau+1}$ only if $\mathsf{cert}_{c,\tau+1}$ is valid and $\mathsf{pk}'_{\tau+1} = \mathsf{pk}_{\tau+1}$.

Theorem 3 (*Security of* NICU3)**.** NICU3 *has both forward and backward key independence, if the given hash function* Hash *is collision-resistant, the signature scheme* SIG *is existentially unforgeable, the function* PRF *is a secure pseudo-random function, and the symmetric encryption scheme* Π *is secure against chosen ciphertext attacks (CCA). More specifically,*

$$\mathsf{Adv}_{\mathsf{NICU3,FW}} \leq \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{coll,Hash}} + \tau \cdot (\epsilon_{\mathsf{euf,SIG}} + 2\epsilon_\Pi^{CCA}) \tag{16}$$

$$\mathsf{Adv}_{\mathsf{NICU3,BW}} \leq \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{coll,Hash}} + (\tau_{max} - \tau) \cdot (\epsilon_{\mathsf{euf,SIG}} + 2\epsilon_\Pi^{CCA}) \tag{17}$$

where $\epsilon_{\mathsf{coll,Hash}}$ *is the probability for any probabilistic polynomial time (PPT) algorithm to find collision of hash function* Hash(), ϵ_Π *the advantage of any PPT algorithm against* Π, ϵ_{PRF} *the advantage of any probabilistic polynomial time (PPT) algorithm against* PRF, *and* $\epsilon_{euf,\mathsf{SIG}}$ *the advantages of any PPT algorithm against* SIG.

Proof. Here we also only provide detailed proof for (16). Let $\mathcal{A}$ be an adversary against NICU3, which can break the forward key independence. Let Adv_n be the advantage of $\mathcal{A}$ in security game Game n. We assume the number of signed messages is polynomial in the security parameter κ and we omit κ in the following proof.

Game 0. This is the original security game in Definition 9, where $\mathcal{A}$ gets all the certificates till time τ (inclusive), all the message–signature pairs $\{m_i, \sigma_i\}$ signed with any $\mathsf{sk}_{\tau-\gamma}$, $0 \leq \gamma \leq \tau$. So we have

$$\mathsf{Adv}_{\mathsf{NICU3,FW}} = \mathsf{Adv}_0 \tag{18}$$

[4] For clarity, we write $\mathsf{CT}_{c,\tau+1}$ separately from the certificate but $\mathsf{CT}_{c,\tau+1}$ can also be set as an extension in the certificate.

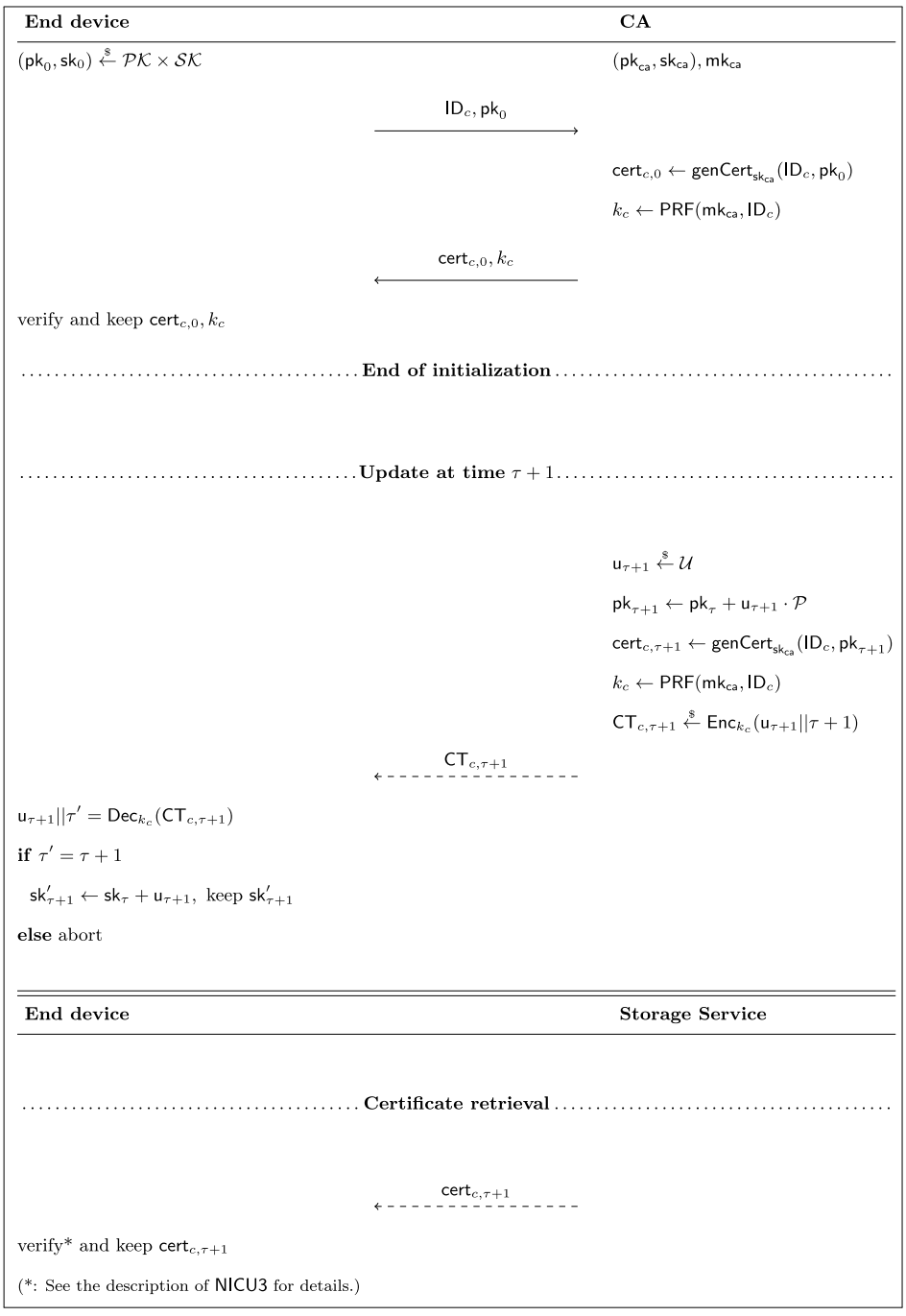

Fig. 6. Non-interactive certificate update protocol NICU3 based on symmetric encryption. Certificate retrieval is included for completeness.

Game 1. We add the following abort rule in this game. If there exist m_i, m_j such that $m_i \neq m_j$ and $\mathsf{Hash}(m_i) = \mathsf{Hash}(m_j)$, abort the game. Note that now Game 1 aborts exactly when a collision of the hash function is found. With a simple probability argument we have

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \epsilon_{\mathsf{coll},\mathsf{Hash}} \tag{19}$$

Game 2. We replace the pseudo-random function with a real random function in this game. If the distribution of $\mathcal{A}$'s output has a non-negligible change in this game compared to in Game 1, we can construct a distinguisher against PRF from $\mathcal{A}$ with advantage bounded by $|\mathsf{Adv}_1 - \mathsf{Adv}_2|$. Therefore we have

$$\mathsf{Adv}_1 \leq \mathsf{Adv}_2 + \epsilon_{\mathsf{PRF}} \tag{20}$$

Game 3. We add the following abort rule in this game. We let the challenger guess the value γ in advance, where γ is the output by $\mathcal{A}$ at the end of this game. If the guess is wrong, abort the game. As the probability of not aborting is at least $\frac{1}{\tau}$, we have

$$\mathsf{Adv}_2 \leq \tau \cdot \mathsf{Adv}_3 \tag{21}$$

Game 4. We replace $\mathsf{CT}_{\tau-\gamma}, \mathsf{CT}_{\tau-\gamma+1}$ with encryption of two random values not equal to $\mathsf{sk}_\tau - \mathsf{sk}_{\tau-1}$ or $\mathsf{sk}_{\tau+1} - \mathsf{sk}_\tau$. If the distribution of $\mathcal{A}$'s output has a non-negligible change in this game compared to in Game 3, we can construct an attacker against Π from $\mathcal{A}$ with advantage $\mathsf{Adv}_3 - \mathsf{Adv}_4$. Since this advantage is also bounded by $2\epsilon_\Pi$, we have

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + 2\epsilon_\Pi \tag{22}$$

The modification in the game above makes $\mathsf{sk}_{\tau-\gamma}$ independent of the update value $\mathsf{u}_{\tau-\gamma}$.

Game 5. This game is unmodified from Game 4, So first we have

$$\mathsf{Adv}_5 = \mathsf{Adv}_4 \tag{23}$$

Now we show how an attacker $\mathcal{F}$ against SIG could be constructed from $\mathcal{A}$. $\mathcal{F}$ is given the public parameters PP, the access to a signing oracle $\mathsf{O}^{\mathsf{SIG}}_{sk*}(\cdot)$ of a SIG instance, and the corresponding public key $\mathsf{pk}*$, as in Definition 2.

To play Game 5 with $\mathcal{A}$ and use $\mathcal{A}$'s output, $\mathcal{F}$ takes the role of the challenger and broadcast the public parameter, the public key of the ca in the security game. $\mathcal{F}$ also generates the encryption–decryption key k_c. Except for time $\tau-\gamma$, $\mathcal{F}$ generates valid signature key pairs $\{(\mathsf{pk}_t, \mathsf{sk}_t)\}, t \leq \tau, t \neq \tau-\gamma$ conforming with the public parameters and give $(\mathsf{pk}_\tau, \mathsf{sk}_\tau)$ to $\mathcal{A}$. The distribution of keys remains the same, though $\mathcal{F}$ does not know $\mathsf{sk}*$.

Note that $\mathcal{F}$ can sign all the necessary messages, either by using the signing keys generated by itself or by querying $\mathsf{O}^{\mathsf{SIG}}_{sk*}(\cdot)$. Thus $\mathcal{F}$ can simulate Game 4 perfectly.

If $\mathcal{A}$ wins Game 5, then the game does not abort and $\mathcal{A}$ outputs some valid (m, σ). So our guess of $\tau-\gamma$ is correct, $\mathsf{SIG.Vfy}(\mathsf{pk}_{\tau-\gamma}, m, \sigma) = 1$, and m must be a new message. As $\mathsf{pk}_{\tau-\gamma} = \mathsf{pk}*$, $\mathcal{F}$ can output (m, σ) as a successful forgery w.r.t $\mathsf{pk}*$. Because the probability of forging signatures for SIG is bounded by euf, SIG, so is the probability for $\mathcal{A}$ to win Game 4. Therefore, we finally have

$$\mathsf{Adv}_4 \leq \epsilon_{\mathsf{euf,SIG}} \tag{24}$$

By combining (18) to (24), the bound in (16) is proved.

4.4. Further security discussion

Security in multi-node-multi-certificate case For simplicity, our current model focuses on the threat faced by each single node associated with only one initial certificate. But in practice, NICU protocols can be deployed over a huge number of nodes, for example, $2^{20} \approx 10^6$, and each device can have multiple active certificates at each time point.

Let us consider that an attacker just wants to forge one signature verifiable with one certificate, without specifying the identity of node or for which certificate. Obviously, the success probability for this attacker is amplified by the total number of possible initial certificates. More specifically, we can directly derive a bound for the attacker's advantage against any secure NICU in this multi-node, multi-certificate case as

$$\mathsf{Adv}_{\mathsf{NICU,FW,MULT}} \leq \delta \cdot \mu \cdot \mathsf{Adv}_{\mathsf{NICU,FW}} \tag{25}$$

$$\mathsf{Adv}_{\mathsf{NICU,BW,MULT}} \leq \delta \cdot \mu \cdot \mathsf{Adv}_{\mathsf{NICU,BW}}, \tag{26}$$

where δ is the total number of nodes sharing the same public parameters, and μ the maximal number of valid certificates owned by one node at one time point. Thus we have to reserve a buffer for the security parameter, if we want every updated certificate to preserve the intended security level. Consequently, it means the length of signatures, ciphertexts and PRF output have to be extended according to the bit length of $\delta \cdot \mu$.

Theoretically, it is possible to reduce or even eliminate the $\delta \cdot \mu$ factor by using tightly secure digital signatures in multi-user setting against corruption. Such signature schemes can be found in [24], but they have not yet been standardized.

5. Performance evaluation

5.1. Test bench and metrics

We implement NICU1, NICU2, NICU3 and conventional certificate update processes in JAVA with the Bouncy Castle Cryptography library 1.64.[5] The CA is simulated with a PC (Ubuntu 16.04, OpenJDK 11.0.1, 4-Core Intel Core i5-4570 4 × 3.2 GHz, Tomcat 8.5.34, MySQL 5.7.26, with 1 Gb wireless LAN access) and a client is simulated with a Huawei Matebook Pro X13 (Windows 10, Intel Core i5-8250U 4 × 1.6 GHz, OpenJDK 11.0.1, with 1 Gb wireless LAN access). The TLS cipher-suite we choose is TLS_ECDHE_ECDSA_WITH_AES_128_GCM _SHA256, which is of high security and used frequently in practice.

For the communication, in the NICU protocols the certificate and (encrypted) $\mathsf{CT}_{c,\tau+1}$ can be pushed to the end device later via another channel. However, for fair comparison, we also consider here the retrieval of this data.

So the major metrics we still use are (1) number of certificates updated per second and (2) communication in Kilobytes. We separate the cases of instantiating ECDSA/EdDSA over different elliptic curves, as computation over the curves may be accelerated differently due to their algebraic properties. We also separate the cases of using a conventional update process with various communication protocols, as the security of messages are different. In Table 1, *Conv./HTTPS Mutual Auth.* means this row is for measuring the performance of a conventional protocol with a TLS connection which enforces that both the client and the server have to authenticate each other with certificates.

5.2. Results summary

As can be observed in Table 1, the certificate update speed of NICU1 is about 3-16 times quicker than that of the conventional protocols (Conv./HTTPS) and requires no communication.

NICU2 pays for asymmetric encryption in the update material so the update speed is only 1-11 times quicker, but the decreased communication overhead remains a clear advantage.

NICU3 stands in the middle of NICU1 and NICU2, as symmetric encryption is much faster than its asymmetric peers. The update speed is 2.8-14 times quicker than conventional ways and the communication advantage is clear.

The results show our new NICU protocols are efficient and we expect to see more performance gain in a cloud-client setting as communication will be the major cost.

Table 2 shows the communication payload of one node to finish the certificate update protocol and to download the different CRLs. (1 yr) and (1 mo) indicates that the certificate will be updated every 1 year and 1 month respectively. We consider here the conventional CRL [8], and the optimized the BfCRL and 2-HCRL solutions [17].

For the time period of 5 years (60 months), 60 new CRLs will be generated. For the verification of peer certificates, (1) in case of BfCRL and CRL, 60 complete CRLs have to be downloaded, (2) in case of 2-HCRL, assume that the node communicates with average 10 peers in one CRL update period.

Compared to the payload to download the CRLs, in all cases (Conv. (1 yr), NICU2 (1 yr) and NICU2 (1 mo)), the payload for the certificate update protocol is negligible. Due to the properties of NICU, we can reduce the certificate update period to the CRL interval. And in this case, no CRL will be required. Although the payload for the certificate update will be 10 times greater (from 7 KiB to 78 KiB), no payload to download the CRLs is needed.

[5] https://www.bouncycastle.org/.

Table 1
Performance of different certificate update protocols. The communication is measured at the client.

	P-256	P-384	P-521	BP256R1	BP384R1	BP512R1	Ed25519	X25519
Performance [#certificates/second]								
NICU1	1183	755	610	937	517	299	2713	2481
NICU2	547	329	271	469	238	135	2013	1966
NICU3	1064	678	546	842	467	270	2398	2240
Conv./HTTP	264	245	207	198	155	92	281	272
Conv./HTTPS	149	148	145	145	118	80	157	149
Conv./HTTPS Mutual Auth.	135	137	132	134	115	76	137	139
Communication [KiB/1000 certificates]								
NICU1/HTTP	1235	1309	1380	1268	1342	1416	1178	1224
NICU2/HTTP	1338	1456	1579	1363	1460	1613	1249	1260
NICU3/HTTP	1289	1391	1482	1318	1403	1517	1262	1290
Conv./HTTP	1709	1833	1977	1740	1977	2011	1589	1628
Conv./HTTPS	3971	4088	4229	3994	4118	4246	3845	3889
Conv./HTTPS Mutual Auth.	5275	5394	5534	5300	5421	5551	5149	5278

Table 2
Communication payload of one node for 5 years, the CRL will be updated in every 1 month. All data are in Megabytes.

	Cert	#revoked certs = 10 K			#revoked certs = 100 K			#revoked certs = 1M		
	Update	2-HCRL	Bf-CRL	CRL	2-HCRL	Bf-CRL	CRL	2-HCRL	Bf-CRL	CRL
Conv. (1 year)	0.009	0.047	0.18	21.700	0.143	1.5	216	0.451	144	2100
NICU2 (1 yr)	0.007	0.047	0.18	21.700	0.143	1.5	216	0.451	144	2100
NICU2 (1 mo)	0.078	0	0	0	0	0	0	0	0	0

6. Conclusion and future work

In this paper, we construct highly efficient and provably secure NICU protocols for certificate based authentication, which can be applied to IoT nodes and CAs. Our analysis shows that NICU1, NICU2 and NICU3 achieve significantly high update speed and can largely reduce bandwidth consumption compared to conventional certificate update protocols.

We would like to work on a more general construction of NICU protocols with fewer restrictions on the properties of the underlying signature scheme. As mentioned before, we do not allow the adversary to obtain all secret keys of a node at time τ, and we have not analyzed the case where the CA is imperfect and each node may have multiple roots of trust, so refining the security framework is also planned. Moreover, we leave it to future research to combine them with other lightweight components for authentication, so that data in IoT can be better protected while preserving efficiency.

CRediT authorship contribution statement

Li Duan: Methodology, Formal analysis, Writing - review & editing. **Yong Li:** Supervision, Formal analysis, Writing - review & editing. **Lijun Liao:** Conceptualization, Validation, Software, Data curation, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank members of applied cryptography research group and reviewers of all versions of this paper for valuable advice and discussions.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2347–2376.

[2] C. Bormann, M. Ersue, A. Keranen, Terminology for Constrained-Node Networks, RFC 7228 (Informational), 2014, URL https://tools.ietf.org/html/rfc7228.

[3] K. Zhao, L. Ge, A survey on the internet of things security, in: 2013 Ninth International Conference on Computational Intelligence and Security, 2013, pp. 663–667, http://dx.doi.org/10.1109/CIS.2013.145.

[4] Y. Li, S. Schäge, Z. Yang, F. Kohlar, J. Schwenk, On the security of the pre-shared key ciphersuites of TLS, in: Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings, 2014, pp. 669–684, http://dx.doi.org/10.1007/978-3-642-54631-0_38.

[5] T. Jager, F. Kohlar, S. Schäge, J. Schwenk, On the security of TLS-dhe in the standard model, in: Advances in Cryptology–CRYPTO 2012, Springer, 2012, pp. 273–293.

[6] Internet Engineering Task Force, Transport Layer Security Version 1.2, RFC 5246, 2008, https://tools.ietf.org/html/rfc5246.

[7] E. Rescorla, Datagram Transport Layer Security Version 1.2, RFC 6347, Obsolete RFC 4347, 2012, URL https://tools.ietf.org/html/rfc6347.

[8] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280 (Proposed Standard), 2008, URL http://www.ietf.org/rfc/rfc5280.txt.

[9] The Federal Office for Information Security, TR-03145-1 secure CA operation, part 1, generic requirements for trust centers instantiating as certification authority (CA) in a public-key infrastructure (PKI) with security level 'high', version 1.1, 2017, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03145/TR03145.pdf.

[10] B. Brecht, D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, R. Goudy, A security credential management system forf v2x communications, IEEE Trans. Intell. Transp. Syst. 19 (12) (2018) 3850–3871.

[11] W. Diffie, M. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory 22 (6) (1976) 644–654.

[12] Z. Shelby, K. Hartke, C. Bormann, The Constrained Application Protocol (CoAP), RFC 7252 (Proposed Standard), 2014, URL https://tools.ietf.org/html/rfc7252.

[13] P. Eronen, H. Tschofenig, Pre-Shared Key Ciphersuites for Transport Layer Security (TLS), RFC 4279 (Proposed Standard), 2005, URL http://www.ietf.org/rfc/rfc4279.txt.

[14] M. Toorani, A. Beheshti, LPKI-A lightweight public key infrastructure for the mobile environments, in: Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on, IEEE, 2008, pp. 162–166.

[15] A. Deacon, R. Hurst, The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments, RFC 5019 (Proposed Standard), 2007, URL http://www.ietf.org/rfc/rfc5019.txt.
[16] G. Rigazzi, A. Tassi, R.J. Piechocki, T. Tryfonas, A. Nix, Optimized certificate revocation list distribution for secure v2x communications, 2017, arXiv preprint arXiv:1705.06903.
[17] L. Duan, Y. Li, L. Liao, Flexible certificate revocation list for efficient authentication in iot, in: Proceedings of the 8th International Conference on the Internet of Things, ACM, 2018, p. 7.
[18] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, C. Wilson, An end-to-end measurement of certificate revocation in the web's PKI, in: Proceedings of the 2015 Internet Measurement Conference, ACM, 2015, pp. 183–196.
[19] M. Mahmoud, K. Akkaya, K. Rabieh, S. Tonyali, An efficient certificate revocation scheme for large-scale ami networks, in: Performance Computing and Communications Conference (IPCCC), 2014 IEEE International, IEEE, 2014, pp. 1–8.
[20] H. Krawczyk, R. Canetti, M. Bellare, HMAC: Keyed-hashing for message authentication, 1997, https://tools.ietf.org/html/rfc2104.
[21] J. Katz, Y. Lindell, Introduction to Modern Cryptography, CRC press, 2014.
[22] W. Wijngaards, Elliptic Curve Digital Signature Algorithm (dsa) for Dnssec, RFC 6605 (Informational), 2012, URL http://buildbot.tools.ietf.org/html/rfc6605.
[23] V. Shoup, Sequences of games: a tool for taming complexity in security proofs., in: IACR Cryptology ePrint Archive 2004, 2004, p. 332.
[24] K. Gjøsteen, T. Jager, Practical and tightly-secure digital signatures and authenticated key exchange, in: Cryptology ePrint Archive, Report 2018/543, 2018, https://eprint.iacr.org/2018/543.

Li Duan received M.Sc. degree in IT-security for network & systems from Ruhr-University Bochum, Germany in 2016. He is currently pursuing Ph.D. degree with Paderborn University, Paderborn, Germany and working as researcher in Huawei Technologies Duesseldorf, Germany. His research interest includes digital signatures and applied cryptographic protocols.

Yong Li received Ph.D. in Applied Cryptography from Ruhr-University Bochum, Germany in 2015. He is currently working as researcher in Huawei Technologies Duesseldorf, Germany. His current research interest includes digital signatures, security model and applied cryptographic protocols.

Lijun Liao received Ph.D. in IT-Security from Ruhr-University Bochum, Germany in 2009. He is currently working as researcher in Huawei Technologies Duesseldorf, Germany. His current research interest includes public key infrastructure and applied cryptographic protocols.