Constructing efficient and secure batch signature schemes

Trieu Quang Phong*, Tran Duy Lai

Institute of Cryptographic Science and Technology, Government Information Security Committee.
*Corresponding author: phongtrieu53@gmail.com.
Received 6 September 2021; Revised 11 December 2021; Accepted 15 December 2021.
DOI: https://doi.org/10.54939/1859-1043.j.mst.csce5.2021.49-60

ABSTRACT

In ordinary signature schemes, such as RSA, DSA, ECDSA, the signing process is performed only for a single message. Due to performance issues, in some contexts, the above solutions will become unsuitable if a party needs to sign multiple messages simultaneously. For example, in the authenticated key exchange protocols based on signatures between client and server, the server is expected to handle multiple key exchange requests from different clients simultaneously. Batch signing is a solution that generates signatures for multi-messages simultaneously with a single (ordinary) signature generation. In this article, we will consider some of the existing batch signing solutions and point out a few of their weakness. To deal with these problems, the paper also proposes two secure types of batch signature schemes, but still ensures the same efficiency as the existing batch signing solution.

Keywords: Merkle Tree; Node; Hash Chain; EUF-CMA Security; Batch Signature Schemes.

1. INTRODUCTION

Digital signatures are very important components in today's electronic transactions. Currently, many digital signature schemes have been standardized, such as ECDSA, GOST R 34.10-2012 (ECRDSA), ECGDSA, etc. These schemes are all present in the international standards ISO/IEC 14888-3 [9].

Normally, the signing process in the ordinary signature schemes is separately done on messages, which requires asymmetric operations that are time-consuming compared to hash operations. These schemes will not be appropriate in some environments where the server needs to provide many signatures per second to the client continuously (such as server-based digital signatures or the authenticated key exchange between the server and the client).

In such cases, a batch signing solution introduced in [4] (and considered applicable in [8] for X-Road [7], which is a data exchange layer for information systems in Estonia) would be much more suitable. This solution enables to simultaneously generate signatures for multiple messages with the same number of asymmetric operations as the generation of a single ordinary signature. The core idea is to create a Merkle hash tree for the messages that need to be signed and then sign its root hash value by using the ordinary signature scheme. The batch signature for each message will then include an ordinary signature on the root of the Merkle tree and a hash chain to prove its participation in the Merkle tree generation. Despite the efficiency, neither [4] nor [8] provides security analysis for this solution.

Unfortunately, we found some limitations on the security of the solution described in [4], [8]. Specifically, the paper will show that the batch signatures in [4], [8] can be forged by using an adaptive chosen message attack. Then, we will propose two alternative batch signature solutions that will be shown to achieve EUF-CMA (**E**xistential **U**n**F**orgeability under adaptive **C**hosen **M**essage **A**ttacks) security.

The rest of this article will be structured as follows: Section 2 will present the preliminaries that will be used. Section 3 presents some of the existing batch signing solutions, including those in [4], [8], and then analyzes their effectiveness. Our contributions will be presented in Section 4, including pointing out the security limitations of the batch signing solution in [4], [8] and then proposing two alternative solutions along with their detailed proofs of security. Finally, section 5 will be the conclusion.

2. PRELIMINARIES

2.1. Merkle Tree

Hash tree: This notion was first introduced by Merkle [1]. Basically, a Merkle hash tree (or Merkle tree) is a special binary tree. The concept of a hash tree means a tree-shaped data structure that consists of nodes, where each node contains a hash value. Formally, the stored data in each (interior) node will be the hash value obtained by hashing the hash values of its child nodes together. The Merkle hash tree is an important component in several signature schemes, such as the batch signature scheme [8], BLT [10], SPHINP+ [11], XMSS [12], etc.

Node: Node is an important part of the Merkle tree structure. There, each node includes information regarding its *hash value*, *child nodes*, and *parent node*. Thus, for a node $node$, we will denote $node$.hash, $node$.left, $node$.right and $node$.parent as its hash value, left child node, right child node, and parent node. For an undefined node, we denote it as NULL.

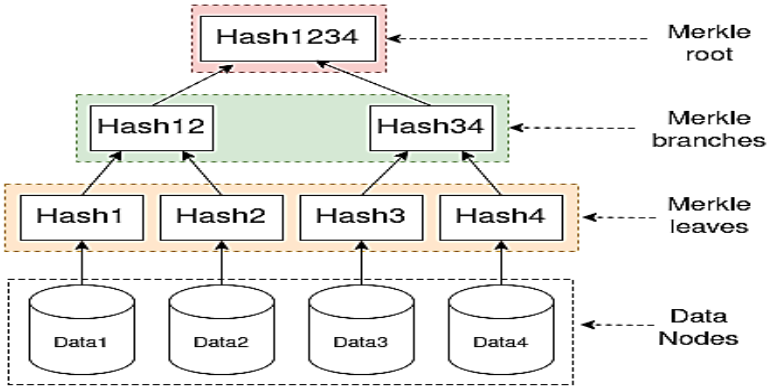

Figure 1. *A Merkle tree with four leaves.*

A Merkle tree is constructed on a hash function $H: \{0,1\}^* \rightarrow \{0,1\}^n$. This structure can be built from a list of leaves $blocks$ by using Alg. 1, and its instance with four leaves is shown in Fig. 1. We will use $\mathcal{T}^H(x_1, \dots, x_\ell)$ to denote a Merkle tree whose ℓ leaves contain ℓ values $x_1, \dots, x_N$ and whose root node is $root$.

Algorithm 1

Input: A list of ℓ leaves $blocks = [blocks[0], block[1], \dots, block[\ell - 1]]$.
Output: The hash linking and the root of Merkle tree corresponding to $blocks$.

1. Defining an intermediate list $nodes$
2. While ($blocks$.size() !=1)
2.1.for (i=0, $j = 0$; i<$blocks$.size(); i=i+2,j++)
2.1.1. If (i!= $blocks$.size()-1)

2.1.1.1. Creating $nodes[j]$ with
$nodes[j]$.hash=H($blocks[i]$.hash.blocks[i+1].hash),
$nodes[j]$.left= $blocks[i]$, $nodes[j]$.right=$blocks[i$+1], $nodes[j]$.parent=NULL.
2.1.1.2. Assigning $blocks[i]$.parent= $blocks[i$+1].parent = $nodes[j]$.
2.1.2. Else
2.1.2.1. Creating nodes[j] with
$nodes[j]$.hash=H($blocks[i]$.hash,$blocks[i]$.hash),
$nodes[j]$.left= $blocks[i]$, $nodes[j]$.right=$blocks[i]$, $nodes[j]$.parent=NULL.
2.1.2.2. Assigning blocks[i].parent = nodes[j].
2.2.Assigning blocks = nodes
2.3.Deleting the list nodes
Assigning the root of hash tree as $root$=blocks[0]

Hash chain: This is a proof to show the participant of a value x in computation of the root hash value $root$. It contains values of all the siblings of the nodes on the unique path from x to the root in the tree. Thus, a hash chain is a list $c = \{(h_1, b_1), \dots, (h_d, b_d)\}$, where each h_i is a hash value n-bits and $b_i \in \{0,1\}$. Based on such a hash chain, we can recover the root hash value $root$ by computing the final element $root = v_d$ of the sequence $\{v_i\}_{i=0}^{d}$ as follows (where $v_0 = x$):

$$v_{i+1} = \begin{cases} H(v_i, h_i) \text{ if } b_i = 1 \\ H(h_i, v_i) \text{ if } b_i = 0 \end{cases}. \quad (1)$$

We will use $x \overset{c}{\rightsquigarrow} root$ and $root \overset{c}{\rightsquigarrow} x$ to denote the computation $root$ from x via the hash chain c. For a hash chain c, its size will be denoted by $|c|$. And, if $c = \{(h_1, b_1), \dots, (h_d, b_d)\}$ then $|c| = d$. Also, for a Merkle tree with ℓ leaves built by Alg. Algorithm 1, the size of hash chain for each leaf is a constant n such that n is the minimal integer number satisfying $2^n \geq \ell$.

2.2. The General Signature Scheme

In this section, we will recall a formal definition (in [3]) of a general signature scheme and some attacks on such a scheme. These definitions are based on [2].

Definition 1 [3]. *A signature scheme* Π *is defined by three algorithms* $(KeyGen_\Pi, SIG_\Pi, V_\Pi)$ *as follows:*

- ***The key generation algorithm*** $\left((pk, sk) \leftarrow KeyGen_\Pi(1^k)\right)$**:** *On input* 1^k*, where* k *is the security parameter, produces a pair* (pk, sk) *of matching public and secret keys.*
- ***The signing algorithm*** $\left(\sigma \leftarrow SIG_\Pi(sk, m)\right)$**:** *Given a message* m *and a pair of public and secret keys* (pk, sk)*, the algorithm Sign produces a signature* σ*.*
- ***The verification algorithm*** $\left(V_\Pi(pk, \sigma, m)\right)$**:** *Given a signature* σ*, a message* m*, and a public key* pk*, the algorithm* V_Π *checks whether* σ *is a valid signature of* m *with respect to* pk*.*

Typically, the security of a signature scheme is considered as the resistance against the adaptive chosen message attacks to find an existential forgery. Specifically, this security guarantee that even if the attacker can adaptively send a list of messages

$m_1, \dots, m_i$ to the signer for getting batch the corresponding signatures $\sigma_1, \dots, \sigma_i$, it is difficult to produce a message $m \neq m_j$ (for $j = 1, \dots, i$) along with its valid fogery. Such security is called as *EUF-CMA*. There are many ordinary signature schemes achieving the EUF-CMA security, such as Schnorr [5], the TEGTSS schemes [6], etc. We can read [5] for a definition of the EUF-CMA security.

Definition 2 [5]. *A signature scheme is secure if an existential forgery is computationally impossible, even under an adaptive chosen message attack.*

3. SOME EXISTING BATCH SIGNATURE SCHEMES

In this section, we'll take a look at some of the existing batch signature solutions, which are constructed by using some ordinary signature scheme.

3.1. Simple Batch Signature with Hash Lists

According to [8], there is a very simple method for converting any conventional signature Π scheme to a batch signature scheme. This scheme is based on the hash lists and works as follows.

In order to create batch signatures for a batch of message $M_1, \dots, M_\ell$, the signer executes:

1. Hashing $m_i = H(M_i)$ for $i = 1, \dots, \ell$.
2. Creating a hash list $L = (m_1, \dots, m_\ell)$ and computing its hash value $m = H(L)$.
3. Signing m by using Π: $\sigma_\Pi = SIG_\Pi(m)$.
4. The signature on M_i is a pair $\sigma_i = (\sigma_\Pi, L)$.

In order to verify a signature (σ_Π, L) on some message M_i, the verifier executes:

1. Computing $m_i = H(M_i)$;
2. Checking whether $m_i \in L$ or not; (Since L is a lits of ℓ blocks with the same length, so it is easy to do this step).
3. Computing $m = H(L)$; and
4. Verifying σ_Π on m by using the verifcation procedure of Π.

It is easy and feasible to implement this scheme if the batch size is relatively small. However, if the batch size is large, this solution is not appropriate, since the size of a batch signature is $\ell \cdot |H| + |\sigma_\Pi|$ (where $|\sigma_\Pi|$ is the size of the ordinary signature and $|H|$ is the number of output bits of H).

3.2. Batch Signature with Merkle Tree

In 1999, Pavlovski and Boyd [4] have introduced a general method for converting a traditional signature scheme Π efficiently to a batch signature scheme by using the Merkle trees [1]. Their *batch signature* $\sigma_1, \dots, \sigma_\ell$ for a batch $M_1, \dots, M_\ell$ are formed as $\sigma_i = (\sigma_\Pi, c_i)$, where σ_Π is a Π signature on the hash value $root$ of the Merkle tree with leaves $M_1, \dots, M_\ell$ and c_i is a hash chain linking M_i to $root$. Besides, this solution was also proposed in [8] by A. Ansper et al. for the X-road system.

The calculation of the Merkle tree only uses hash computations. Therefore, creating a batch signature is almost as efficient as the generation of a single ordinary signature. Also, in this case, the size of a batch signature is $|H| \cdot \log(\ell) + |\sigma_\Pi|$ which is shorter than one with hash lists. An example of signing for a batch with four messages is illustrated in Fig. 2.

The verification procedure for a signature (σ_Π, c_i) on message M_i (illustrated in Fig.

3) includes two steps: (1) recovering the root hash value $root$ from M_i by using c_i; and (2) verifying σ_Π on $root$ by using the verification procedure of Π.

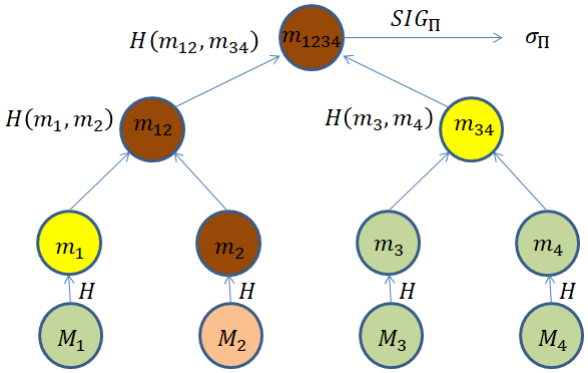

Figure 2. *The signing procedure for a batch (M_1, M_2, M_3, M_4) based on the Merkle tree. The signature on M_2 is $\sigma_2 = (\sigma_\Pi, c_2) = (\sigma_\Pi, \{(m_1, 0), (m_{34}, 1)\})$, where $root = m_{1234}$.*

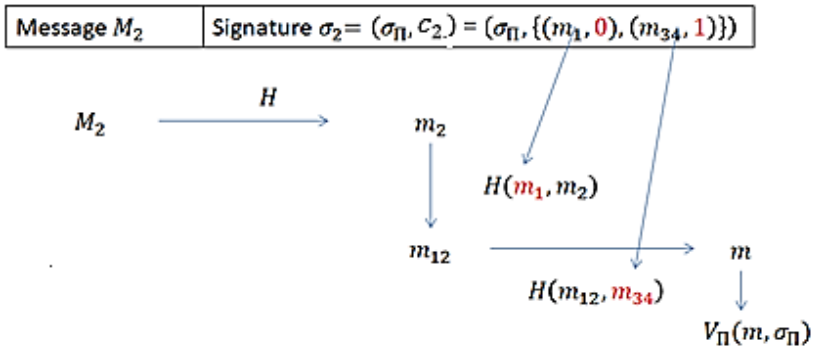

Figure 3. *The verification procedure for a signature $(\sigma_\Pi, c_2) = (\sigma_\Pi, \{(m_1, 0), (m_{34}, 1)\})$ on message M_2.*

Efficiency. We can see that in order to sign ℓ messages simultaneously by using the above method, the server only executes one ordinary signing process and compute about $2\ell - 1$ the hashing computations instead of performing ℓ ordinary signing processes. Since the asymmetric cryptography operations used in the ordinary signing process are much more expensive than the hashing operations, the batch signature scheme in [4] and [8] is an effective solution for generating signatures. However, this solution makes the verification process increases about $1 + \log_2 \ell$ hash operations.

In table 1, we present the experimental results of comparing the signing and verification performance between the RSA signature scheme and the batch signature scheme (based on the Merkle tree and RSA signature scheme). These results are in agreement with the theoretical analysis that the batch signing solution is significantly more efficient than the ordinary one, while the time of its verification process only increases by a relatively small amount. Also, in some applications, such as the authenticated key exchange protocols based on signatures between a server and many clients, the verification process for batch signatures would be only performed by multiple clients and thus does not increase the computational requirement to the server-side. Therefore, as long as the client is still using regular signatures, the server's cost of verifying the client signature does not change.

Table 1. *The experimental results of comparing the signingand verification performance between the RSA signature scheme and the batch signature scheme (based on Merkle tree and RSA signature scheme).*

	The batch signature based on Merkle tree and RSA signature scheme		RSA signature scheme	
	Signing time	Verification time	Signing time	Verification time
10 messages	0.022 s	0.069 s (Average: 0.0069 s)	0.279 s	0.069 s (Average: 0.0069 s)
100 messages	0.05 s	0.71 s (Average: 0.0071 s)	2.57 s	0.70 s (Average: 0.0070 s)
1000 messages	0.75 s	7.22 s (Average: 0.0072 s)	25.87 s	6.93 s (Average: 0.0069 s)

3. PROPOSING SOME SECURE BATCH SIGNATURE SCHEMES

Although the batch signing solution described in Section 3.2 is efficient in generating signatures, there is a limitation of its security. Therefore, we will propose two types of secure batch signature schemes by using the size of the hash chain component in a batch signature as part of the verification process. Note that our proposals are added some relatively small computations compared to the batch signature scheme in [4], [8], thus retaining the efficiency of the original solution essentially.

3.1. The Weakness of the Existing Batch Signing Solution based on the Merkle Tree

As mentioned above, we now point out that the existing batch signature scheme based on the Merkle tree does not achieve the EUF-CMA security defined in Definition 2. Indeed, the attacker can easily produce a valid forgery on an arbitrary message M as follows:

i. Firstly, the attacker hash M to obtain $H(M)$ and then randomly chooses a bit string h which has the same length as $H(M)$.

ii. Next, the attacker sends $H(M)\|h$ as his query to the signing oracle for getting back a batch signature (σ_Π, c).

iii. Finally, the attacker returns (σ_Π, c'), where $c' = (h, 1)\|r$, as a forgery on M.

Since $H(M) \overset{(h,1)}{\rightsquigarrow} H(H(M), h) = H(H(M) \parallel h)$, and (σ_Π, c) is a valid batch signature on $H(M)\|h$, so we also have (σ_Π, c') is a valid batch signature on M. On the other hand, since the attacker has never sent message M as a request to the signing oracle, he is considered to be successful in finding an existential forgery without breaking the underlying signature scheme as well as any security assumption of the hash function H. In other words, such a batch signature scheme does not achieve the EUF-CMA security defined in definition 2.

For more clarity, we can imagine the attacker's works through Fig. 4. In which, to produce a valid forgery on some message M_2, the attacker chooses a message $M_2^* = m_2^* \parallel h$ (where $m_2^* = H(M_2)$) and requires the signing oracle to create a signature on M_2^*.

As a result of the example of signing a batch with four messages, we obtain $\sigma_2 = (\sigma_\Pi, c_2) = (\sigma_\Pi, \{(m_1, 0), (m_{34}, 1)\})$ as a batch signature on M_2^*. Then, it is easy for the attacker to produce $\sigma_2^* = (\sigma_\Pi, c_2^*) = (\sigma_\Pi, \{(h, 1), (m_1, 0), (m_{34}, 1)\})$ as a batch signature on M_2. This signature is valid on message M_2 since

$$H\big(H\big(m_1, H(H(M_2), h)\big), m_{34}\big) = H\big(H\big(m_1, H(M_2^*)\big), m_{34}\big) = m_{1234}$$

and σ_Π is a valid signature on message m_{1234} in the scheme Π. Thus, the attacker is considered successful when finding an existential forgery.

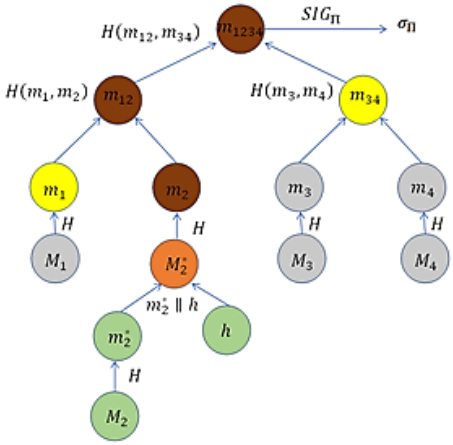

Figure 4. *The adaptive chosen message M_2^* in the case of signing a batch with 4 messages.*

3.2. Proposal 1

It is easy to see that if the size of the hash chain component in a batch signature is a constant len and then it is used as part of the verification process. That is, we need to check whether the size of the hash chain component in the verified signature is equal to len or not. In that way, the attacker will no longer produce a valid existential forgery by using a trivial method discussed in section 4.1 since the forged batch signature does not pass the size check step. Below, we will present a secure batch signature by using the above idea.

For a traditional signature scheme $\Pi = (KeyGen_\Pi, SIG_\Pi, V_\Pi)$, we will construct a batch signature scheme Π′ with three key generation, signing and verification algorithm as follows:

- *The key generation:*
  (1) Using $KeyGen_\Pi$ to generate a key pair (sk_Π, pk_Π).
  (2) Setting the secret key $sk = sk_\Pi$ and the public key $pk = (pk_\Pi, len)$, where len is a parameter indicating the height of the Merkle tree that will be built in the signing process. Also, the public key $pk = (pk_\Pi, len)$ specifies that the user can sign at most 2^{len} messages simultaneously. Thus, the selection of the parameter len will depend on the need to simultaneously sign multi-messages, such as if a user usually signs a single message, then he can register his public key as $pk = (pk_\Pi, 0)$.
- *The signing algorithm:* In order to sign a batch $\{M_1, \dots, M_\ell\}$, where $\ell \leq 2^{len}$, the signer executes:

(1) Computing $m_i = H(M_i)$, for $i = 1, \dots, \ell$.

(2) Building a Merkle tree $\mathcal{T}^H$ for 2^{len} values $\left\{m_1, \dots, m_\ell, \underbrace{m_\ell, \dots, m_\ell}_{2^{len}-\ell}\right\}$, and assuming that its root hash value is $root$.

(3) Using SIG_Π to generate a signature σ_Π on $root$.

(4) Extracting ℓ has chains $c_1, \dots, c_\ell$ for ℓ first leaves of the Merkle tree $\mathcal{T}^H$.

(5) Returning (σ_Π, c_i) as a signature on M_i, for $i = 1, \dots, \ell$.

- *The verification algorithm:* For a signature (σ_Π, c), a message M, and the public key $pk = (pk_\Pi, len)$, the verifier executes:

(1) Checking whether the size of c is equal to len or not. If they are different, then the signature is not valid.

(2) Computing $m = H(M)$, and $root \overset{c}{\rightsquigarrow} m$.

(3) Using V_Π to check the validity for the pair $(root, \sigma_\Pi)$. If it holds then the signature is valid, otherwise the signature is not valid.

We will call the above transformation from Π to Π' as $Trans$, i.e. $\Pi' = Trans(\Pi)$. Below, we show the result about the EUF-CMA security of $\Pi' = Trans(\Pi)$ in the random oracle model. It is worthy to note that the number of signing queries from the attacker to Π' and the response time for these queries are negligible compared to the output space of the hash function H.

Proposition 1. *If a signature scheme Π is EUF-CMA secure (according to Definition 2) and the hash function is modeled as a random oracle, then the batch signature scheme $\Pi' = Trans(\Pi)$ is also secure in the same sense.*

Proof. Let $\mathcal{A}$ be an attacker that tries to find an existential forgery and can adaptively send q_s signing queries to Π' for getting back signatures on q_s messages $M_1, \dots, M_{q_s}$. We assume that the signing processes for q_s above message yield t Merkle trees with the corresponding root hash values $root_1, \dots, root_t$. Note that since messages $M_1, \dots, M_{q_s}$ are possibly signed in different batch signing processes, the number of Merkle trees that are built in these processes cannot exceed the number of signing messages requested, i.e., $t \leq q_s$.

Then, if $\mathcal{A}$ can produce a message $M^* \neq M_i$ $(\forall i \in \{1, \dots, q_s\})$ and its valid signature, assuming that it is $\sigma^*_{\Pi'} = (\sigma^*_\Pi, c^*)$, there are two following cases:

- Case 1: if $M^* \overset{c^*}{\rightsquigarrow} root^* \neq root_i, i \in \{1, \dots, t\}$, it implies that $\mathcal{A}$ can produce a valid signature σ^*_Π on "new" message $root^*$which Π has never signed. Therefore, (σ^*_Π, c^*) is a successfully existential forgery, which is contrary to the assumption of the EUF-CMA of the scheme Π.
- Case 2: if $\exists i \in \{1, \dots, t\}$ such that $root^* = root_i$, it implies that σ^*_Π has been created by Π on $root^*$. Assuming that $M \in \{M_1, \dots, M_{q_s}\}$ is the message which its batch signature is (σ^*_Π, c) and the corresponding Merkle tree has the root hash value $root_i$. Since c^* and c has the same length len, we can assume that $\{v_j\}_{j=0}^{len}$ is the sequence

in computations from $H(M)$ to $root_i$ via c; and $\{v_j^*\}_{j=0}^{len}$ is the sequence in computations from $H(M^*)$ to $root^*$ via c^*. On the other hand, since $M \neq M^*$ and $H(M^*) \overset{c^*}{\rightsquigarrow} root^* = root_i \overset{c}{\rightsquigarrow} H(M)$, there is $j \in \{0, \dots, len-1\}$ such that $v_j \neq v_j^*$ and $v_{j+1} = v_{j+1}^*$. This result yields a collision of H. We note that there are q_s messages, and for each message, there are at most len positions in which a collision may appear. Thus, the probability that Case 2 occurs is at most $\frac{q_s \cdot len}{2^{|H|}}$.

According to the above cases, the probability that $\mathcal{A}$ is successful in finding an existential forgery in the scheme Π' under the adaptive chosen message attacks is negligible. ■

Thus, we have presented a general batch signature scheme built from an ordinary one. Also, we proved that our proposed scheme is EUF-CMA secure as long as the underlying scheme achieves this security, such as Schnorr [5], or TEGTSS [6], etc.

3.3. Proposal 2

Although our first proposal provides a family of secure batch signature schemes, such schemes are only suitable for applications with a stable number of requests. Indeed, assuming a server is configured to handle 2^{10} requests for each signature generation, we would have to generate a hash tree with 2^{10} leaves even if there is only one message to be signed. In that case, our first proposal is less efficient than its underlying signature scheme. Therefore, this section proposes another type of secure batch signature scheme that minimizes the above problems.

For a traditional signature scheme $\Pi = (KeyGen_\Pi, SIG_\Pi, V_\Pi)$, we will construct a batch signature scheme Π' with three key generation, signing and verification algorithm as follows:

- *The key generation:*
  (1) Using $KeyGen_\Pi$ to generate a key pair (sk_Π, pk_Π).
  (2) Setting the secret key $sk = sk_\Pi$ and the public key $pk = pk_\Pi$.
- *The signing algorithm:* In order to sign a batch $\{M_1, \dots, M_\ell\}$, where $\ell \leq 2^{len}$, the signer executes:
  (1) Computing $m_i = H(M_i)$, for $i = 1, \dots, \ell$.
  (2) Building a Merkle tree $\mathcal{T}^H$ for ℓ values $\{m_1, \dots, m_\ell\}$, and assuming that its root hash value is $root$.
  (3) Finding n such that n is the smallest integer number satisfying $2^n \geq \ell$.
  (4) Using SIG_Π to generate a signature σ_Π on $root \parallel n$.
  (5) Extracting ℓ has chains $c_1, \dots, c_\ell$ for ℓ first leaves of the Merkle tree $\mathcal{T}^H$.
  (6) Returning (σ_Π, c_i) as a signature on M_i, for $i = 1, \dots, \ell$.
- *The verification algorithm:* For a signature (σ_Π, c), a message M, and the public key $pk = (pk_\Pi, len)$, the verifier executes:
  (1) Extracting the size of c, assuming that it is equal to n.
  (2) Computing $m = H(M)$, and $root \overset{c}{\rightsquigarrow} c$.

(3) Using V_Π to check the validity for the pair $(root \parallel n, \sigma_\Pi)$. If it holds then the signature is valid, otherwise the signature is not valid.

We will call the above transformation from Π to Π′ as $Trans^*$, i.e. $\Pi' = Trans^*(\Pi)$. Below, we show the result about the EUF-CMA security of $\Pi' = Trans^*(\Pi)$ in the random oracle model.

Proposition 2. *If a signature scheme Π is EUF-CMA secure (according to Definition 2) and the hash function is modeled as a random oracle, then the batch signature scheme $\Pi' = Trans^*(\Pi)$ is also secure in the same sense.*

Proof. Let $\mathcal{A}$ be an attacker that tries to find an existential forgery and can adaptively send q signing queries to Π′ for getting back signatures on q messages $M_1, \dots, M_q$. We assume that the signing processes for q above message yield t Merkle trees with the corresponding root hash values $root_1, \dots, root_t$ and the corresponding heights $d_1, \dots, d_t$. It is similar to Proposition Proposition **1**, we have $t \le q$. Also, the size of a hash chain linking some leaf to the root of the i-th tree is d_i, for $i = 1, \dots, t$. Let $d = \max_{i \in \{1,\dots,t\}}\{d_1, \dots, d_t\}$.

Then, if $\mathcal{A}$ can produce a message $M^* \neq M_i$ $(\forall i \in \{1, \dots, q\})$ and its valid signature, assuming that it is $\sigma^*_{\Pi'} = (\sigma^*_\Pi, c^*)$, there are two following cases:

- **Case 1:** if $M^* \overset{c^*}{\rightsquigarrow} root^* \neq root_i$ or $|c^*| \neq d_i$, $i \in \{1, \dots, t\}$, it implies that $\mathcal{A}$ can produce a valid signature σ^*_Π on "new" message $root^* \parallel |c^*|$ which Π has never signed. Therefore, (σ^*_Π, c^*) is a successfully existential forgery, which is contrary to the assumption of the EUF-CMA of the scheme Π.
- **Case 2:** if $\exists i \in \{1, \dots, t\}$ such that $root^* = root_i$ and $|c^*| = d_i$, it implies that σ^*_Π has been created by Π on $root^* \parallel |c^*| = root_i \parallel d_i$. Assuming that $M \in \{M_1, \dots, M_q\}$ is the message which its batch signature is (σ^*_Π, c) and the corresponding Merkle tree has the root hash value $root_i$. Since c^* and c has the same length d_i, we can assume that $\{v_j\}_{j=0}^{d_i}$ is the sequence in computations from $H(M)$ to $root_i$ via c; and $\{v_j^*\}_{j=0}^{d_i}$ is the sequence in computations from $H(M^*)$ to $root^*$ via c^*. On the other hand, since $M \neq M^*$ and $H(M^*) \overset{c^*}{\rightsquigarrow} root^* = root_i \overset{c}{\rightsquigarrow} H(M)$, there is $j \in \{0, \dots, d_i - 1\}$ such that $v_j \neq v_j^*$ and $v_{j+1} = v_{j+1}^*$. This result yields a collision of H. We note that there are q messages, and for each message, there are at most d positions in which a collision may appear. Thus, the probability that Case 2 occurs is at most $\frac{q \cdot d}{2^{|H|}}$.

According to the above cases, the probability that $\mathcal{A}$ is successful in finding an existential forgery in the scheme Π′ under the adaptive chosen message attacks is negligible. ■

Thus, we also obtain that the second proposal of batch signature scheme achieves the EUF-CMA security. Obviously, this proposal is more flexible to use than our first proposal.

4. CONCLUSION

In this article, we have considered several types of batch signature schemes that may be suitable in environments where the server needs to efficiently provide multiple

signatures to clients. There, the batch signature schemes based on the Merkle tree have an advantage in the size of the batch signature. In this case, a batch signature on a message m is the combination of a hash chain to recover the root hash value $root$ from m and an ordinary signature on $root$. However, there are no constraints on the size of the hash chain component of a batch signature in such schemes. According to this observation, we have shown that the existing batch signature schemes, which are based on the Merkle tree, will not achieve the UEF-CMA security.

By adding some constraints regarding the size of the hash chain component of a batch signature, we have proposed two types of secure batch signature schemes built from a secure ordinary signature scheme (e.g., Schnorr [5], or TEGTSS [6], etc.). In the first approach, we propose to use an additional constant in the public key that specifies the height of the hash tree built in a batch signature generation, and then it will be used as part of the verification process. However, this approach will not be suitable in cases where the number of requests to a server is significantly smaller than its capacity configuration. Thus, in the second approach, we only build a hash tree whose height is based on the number of messages to be simultaneously signed, and its height and root will be signed together. By using such a way, our second proposal is more flexible to use than the first one.

Acknowledgment: *We would like to thank my co-workers at the Institute of Cryptographic Science and Technology Government Information Security Committee for their valuable and constructive suggestions during the planning and development of this research work. Also, their support is the motivation to help us complete this work.*

REFERENCES

[1]. R. C. Merkle. "*Protocols for public-key cryptosystems*". In: Proceedings of the 1980 IEEE Symposium on Security and Privacy, pp. 122–134 (1980).

[2]. S. Goldwasser, S. Micali, and R. Rivest. "*A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks*". SIAM Sournal of Computing, 17(2):281-308, April 1988.

[3]. D. Pointcheval, and J. Stern. "*Security proofs for signature schemes*". International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1996.

[4]. Christopher J. Pavlovski and Colin Boyd. "*Efficient batch signature generation using tree structures*". In International Workshop on Cryptographic Techniques and E-Commerce: CrypTEC'99, pages 70–77. City University of Hong Kong Press, 1999.

[5]. D. Pointcheval, and J. Stern. "*Security arguments for digital signatures and blind signatures*". Journal of cryptology 13.3 (2000): 361-396.

[6]. Ernest F. Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung, "*Design validations for discrete logarithm based signature schemes*", In Hideki Imai and Yuliang Zheng, editors, PKC 2000, volume 1751 of LNCS, pages 276-292. Springer-Verlag, 2000.

[7]. A. Kalja. "*The first ten years of X-road*". In Estonian Information Society Yearbook 2011/2012, pages 78–80. Department of State Information System, Estonia, 2012.

[8]. A. Ansper, et al. "*High-performance qualified digital signatures for X-road*". In Nordic Conference on Secure IT Systems. Springer, Berlin, Heidelberg, 2013.

[9]. International standard ISO/IEC DIS 14888-3. *Information technology -- Security techniques -- Digital signatures with appendix -- Part 3: Discrete logarithm based mechanisms*. ISO/IEC JTC 1/SC 27, 2016.

[10]. A. Buldas, R. Laanoja, and A. Truu. "*A server-assisted hash-based signature scheme*". In Nordic Conference on Secure IT Systems. 2017. Springer.

[11].D. J. Bernstein, et al. "*The SPHINCS+ signature framework*". Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019.

[12].J. W. Bos, et al. "*Rapidly verifiable XMSS signatures*". IACR Transactions on Cryptographic Hardware and Embedded Systems (2021): 137-168.

TÓM TẮT

XÂY DỰNG CÁC CHỮ KÝ BÓ HIỆU QUẢ VÀ AN TOÀN

Trong các lược đồ chữ ký thông thường như RSA, DSA, ECDSA,… quá trình ký chỉ thực hiện đối với từng thông điệp đơn lẻ. Do vấn đề về hiệu suất, trong một số ngữ cảnh những giải pháp kể trên sẽ trở nên không phù hợp nếu một bên cần ký đồng thời nhiều thông điệp, chẳng hạn đối với các giao thức trao đổi khóa có xác thực dựa trên chữ ký giữa máy khách-máy chủ và máy chủ được mong muốn xử lý nhiều yêu cầu trao đổi khóa cùng lúc từ nhiều máy khách khác nhau. Ký bó là một giải pháp hiệu quả giúp tạo ra đòng thời các chữ ký cho một loạt thông điệp chỉ với một lần tạo chữ ký thông thường. Trong bài báo này, chúng tôi sẽ xem xét một số giải pháp ký bó hiện có và chỉ ra một vài nhược điểm của chúng. Để cải thiện những vấn đề đó, bài báo cũng đề xuất 02 dạng lược đồ chữ ký bó an toàn nhưng vẫn đảm bảo độ hiệu quả như của giải pháp ký bó đã có.

Từ khoá: Cây băm Merkle; Nút; Chuỗi băm; Độ an toàn EUF-CMA; Chữ ký bó.