Hindawi Publishing Corporation
Mobile Information Systems
Volume 2015, Article ID 901418, 12 pages
http://dx.doi.org/10.1155/2015/901418

Research Article

Secure Mobile Agent from Leakage-Resilient Proxy Signatures

Fei Tang,[1,2] Hongda Li,[1,2] Qihua Niu,[1,2] and Bei Liang[1,2]

[1]*The Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, No. 89 Minzhuang Road, Haidian District, Beijing 100093, China*
[2]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, No. 89 Minzhuang Road, Haidian District, Beijing 100093, China*

Correspondence should be addressed to Fei Tang; tangfei127@163.com

Received 27 February 2014; Accepted 3 March 2014

Academic Editor: David Taniar

A mobile agent can sign a message in a remote server on behalf of a customer without exposing its secret key; it can be used not only to search for special products or services, but also to make a contract with a remote server. Hence a mobile agent system can be used for electronic commerce as an important key technology. In order to realize such a system, Lee et al. showed that a secure mobile agent can be constructed using proxy signatures. Intuitively, a proxy signature permits an entity (delegator) to delegate its signing right to another entity (proxy) to sign some specified messages on behalf of the delegator. However, the proxy signatures are often used in scenarios where the signing is done in an insecure environment, for example, the remote server of a mobile agent system. In such setting, an adversary could launch side-channel attacks to exploit some leakage information about the proxy key or even other secret states. The proxy signatures which are secure in the traditional security models obviously cannot provide such security. Based on this consideration, in this paper, we design a leakage-resilient proxy signature scheme for the secure mobile agent systems.

1. Introduction

Mobile agents [1–3] are designed as some autonomous software entities which are able to sign some messages in a remote server on behalf of a customer without exposing its secret key. Therefore, a mobile agent system can be used for electronic commerce in many ways such as negotiating something with other entities, searching and buying special products or services on behalf of a customer, and selling products on behalf of a shopping server. As shown by previous works, a mobile agent system can be constructed using some proxy signature schemes; for example, Lee et al. [4] used a strong nondesignated proxy signature scheme; they also provided an RSA-based and Schnorr-based constructions of secure mobile agent.

Proxy Signatures. This notion was first introduced by Mambo et al. [5] in 1996. In a proxy signature scheme, an entity called delegator may delegate its signing right to another entity called proxy who can then sign some specified messages on behalf of the delegator; we call such signatures as proxy signatures. Finally, the verifier can be convinced from the proxy signatures that the original signer's agreement on the signed message and such proxy signatures must be computed by the proxy rather than the delegator. Obviously, proxy signatures are very useful in many application scenarios, for example, mobile agents [3, 6–9] and mobile communications [10, 11]. In the existing proxy signature schemes, the model of delegation by warrant [5] (a signed warrant, e.g., $W := ID_{\text{proxy}} \parallel \mathcal{M} \parallel$ indate $\parallel \cdots$, used to describe the validity of the delegation) has received the most attention. Kim et al. [12] suggested that a proxy key should be generated from such warrant. After Mambo et al.'s seminal work, many variants or improved schemes have been proposed (e.g., see [4, 11, 13–17]).

BPW Transformation. Boldyreva et al. [13] (henceforth called BPW) have given a secure generic construction of proxy schemes in the model of delegation by warrant from any secure ordinary signature scheme. Informally, to generate a proxy key, the original signer first signs a concatenation of

the proxy's public key and a warrant with a specific way to obtain a delegation certificate. Then the proxy could set up the proxy key by himself using this delegation certificate. Finally, the proxy could sign some messages that are described in the warrant on behalf of the original signer (cf. Section 4 of [13] for detailed description).

Multilevel Proxy Model. Malkin et al. [14] extended the general proxy signatures to the scenario of multilevel proxy, where the proxy can also delegate the proxy signing right to another proxy (in such setting the former proxy also is a delegator); similarly, the second proxy also can delegate its proxy signing right to another, and so on. We call the identities that the original signer and all proxies construct a delegation chain, that is, (`original signer`)-(1th `proxy`)-(2th `proxy`)-$\cdots$-(jth `proxy`)-$\cdots$.

Security Models for Proxy Signatures. Due to the additional property of the proxy signatures, how to define the security for the proxy signatures is more complicated than the standard signatures [18]. In [19], Mambo et al. introduced several security notions (then enhanced by Lee et al. [4]) for the proxy signatures (here we omit them; please refer to [4, 19] for detailed description). These notions provide some intuitive security requirements for the proxy signatures, but corresponding security definitions are unclear (i.e., lacking of formal definitions), so many constructions were shown to be insecure and then fixed and finally to be shown insecure again (e.g., [4, 19, 20]). Subsequently, Boldyreva et al. [13] first presented a well-defined security model for the proxy signatures. In their model, the adversary is allowed to corrupt an arbitrary number of users and learn their secret keys. Moreover, the adversary can also register some public keys on behalf of new users. Then, the adversary interacts with honest users playing the role of a delegator or a proxy and it can see the transcripts of all executions of the delegation protocol between the honest users. It is a rather strong security model. Malkin et al. [14] later extended this model to allow multilevel proxy signatures; they also showed that proxy signatures are equivalent to key-insulated signatures [21]. The models of [13, 14] both are registered key models, which means that it is required that the adversary submits the secret and public keys of all users used in the model except a single challenging user. Schuldt et al. [15] got rid of this requirement and gave a new security model, existential unforgeability under adaptive chosen message attack with proxy key exposure (EU-CMA-PKE). In this model, adversary directly controls all user's secret keys of the delegation chain except the challenging user; furthermore, the adversary can corrupt some user to obtain the proxy keys (see Section 4 of [15] for more detailed description).

Black-Box Assumption versus Reality. In the security model of cryptographic schemes, traditionally, it is assumed that the secret internal state (secret key, randomness, etc.) of the schemes is completely hidden to the adversary, and hence the adversary in the traditional black-box model only can access an oracle to learn the input and output behaviors about the scheme. Unfortunately, many cryptographic engineers have shown that this assumption is not true in real world applications. They have designed a large class of realistic attacks, called side-channel attacks, to detect some leakage information about the secret state, for example, timing attacks [22], power consumption [23], and fault attacks [24, 25]. Therefore, if we implement a mobile agent system from a secure proxy signature that is in the traditional security model, it may be also insecure if the device of mobile agent encounters the side-channel attacks.

Leakage-Resilient Cryptography. To resist such side-channel attacks, cryptographers have proposed many countermeasures in the past few years. Leakage-resilient cryptography is one of them, which means that a cryptosystem is also secure; even the adversary obtains some bounded (even arbitrary) leakage information about the secret internal state.

To model the security of cryptographic schemes in the leakage-resilient cryptography setting with a formal way,considering an adversary attacks a scheme besides the ordinary queries (as in the black-box model), it also can adaptively choose arbitrary polynomial time computable functions (named leakage functions) $f_i : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ to obtain some information about the secret internal state. The restrictions of the input and output for such leakage functions depend on the leakage models. Here, we briefly present some of them.

(i) Only computation leaks model, introduced by Micali and Reyzin [26]: in this model, leakage is assumed to only occur on values that are currently accessed during the computation. Therefore, the input of the leakage function f_i is confined to the *active* part of the internal secret state, while the passive part of the secret state is not taken as input to the leakage function.

(ii) Bounded leakage model: the overall amount of the leakage should be bounded on a prespecified value λ.

(iii) Continual-leakage model, introduced by Brakerski et al. [27] and Dodis et al. [28], independently: in this model, the secret key is allowed to be refreshed, while the corresponding public key remains fixed. Then the amount of the leakage is bounded only in between any two successive key refreshes and the overall amount can be unbounded.

Many cryptographic schemes have been proposed in the leakage-resilient cryptography setting based on different leakage models, for example, leakage-resilient stream ciphers [29], leakage-resilient zero knowledge [30], leakage-resilient PKE [31, 32], leakage-resilient IBE [33, 34], and leakage-resilient signatures [35–40].

Leakage-Resilient Signatures. In this paper, we focus on the construction of leakage-resilient signature schemes. Alwen et al. [35] gave a construction of leakage-resilient signature scheme in the random oracle model which may tolerate leakage of up to half the secret key. Then Katz and Vaikuntanathan [38] constructed a bounded leakage-resilient signature scheme in the standard model which can tolerate leakage

of up to $\ell - \ell^{\epsilon}$ (ℓ denotes the bit-length of the secret key) bits of information about the secret key. In the same paper, they also introduced the notion of fully leakage-resilient signatures which means that it is EU-CMA secure even the adversary may obtain leakage information on all internal state values that are used throughout the lifetime of the scheme. Boyle et al. [36] then improved their scheme to a full one which can be resilient to any leakage of length $(1 - o(1))\ell$ bits. Faust et al. [37] constructed a tree-based leakage-resilient signature scheme (in the model of "only computation leaks") which can be instantiated with any 3-time bounded leakage-resilient signature. Their scheme resilient to $\lambda = \lambda'/3$ bits per signing process, where λ' is size of the underlying 3-time signature scheme, can leak in total.

Our Contribution. Proxy signatures are often proposed for use in applications where signing is done in a potentially hostile environment; for example, if we use a proxy signature to realize a mobile agent system, then the proxy key is stored in a laptop, or even an IC card, which might become infected by malware. In such setting, an adversary who launches side-channel attacks can detect some leakage information about the proxy key or even other internal states. Based on this consideration, we construct a proxy signature scheme in the setting of leakage-resilient cryptography, the leakage-resilient proxy signature (LRPS), for the first time. The proposed LRPS scheme maintains the properties of these two primitives, leakage-resilient cryptography and proxy signatures.

To define the security notion to the LRPS scheme, we combine the existing security models of proxy signatures and leakage-resilient cryptography to put forward the security model of existential unforgeability against the adaptive chosen message and leakage attacks (EU-CMLA (We also introduce the notion of EU-CMLA-PKE which is extended from EU-CMA-PKE in [15] for the full construction of the LRPS in Appendices.)). Furthermore, we also construct a concrete LRPS scheme under the delegation by warrant and multilevel proxy models, it can be regarded as a concrete implementation of the BPW transformation in the setting of leakage-resilient cryptography. We use a tree-based signature scheme to construct the proxy signature scheme, which is different than the method that [13, 15] adopted; they both adopted an aggregate signature [41]. Hence our construction provides an alternative method to the construction of the proxy signatures. The concrete construction of the LRPS scheme is based on Faust et al.'s [37] (henceforth called FKPR, in TCC 2010) leakage-resilient signature scheme.

2. Definitions

In this section, we present some basic definitions for this paper: the notion of the stateful signatures and its security in the black-box model and in the presence of leakage, respectively.

2.1. Notations. 1^k denotes the string of k ones for $k \in N$. $|x|$ denotes the length of the bit string x if x is a bit string; $|S|$ denotes the number of the entries in the set S. $s \xleftarrow{\$} S$ means randomly choosing an element s from the set S. We write $y \leftarrow \mathsf{A}(x)$ to indicate that running the algorithm A with input x and then outputs y and $y \xleftarrow{\$} \mathsf{A}(x)$ has the same indication except that A is a probabilistic algorithm. We use the notation $s_1 \parallel s_2$ to denote the concatenation of the bit strings s_1 and s_2; if they are not strings, we assume that they will be encoded as a string before the concatenation takes place. Lastly we write PPT for the probabilistic polynomial time.

2.2. Stateful Signatures. A signature scheme SIG consists of three algorithms, key generation, signing, and verification denoted by Kg, Sign, and Vfy, respectively. We say that a signature scheme is stateful if the Sign algorithm is stateful, which means that the secret key will be refreshed after (or before) each signing process, while its corresponding public key remains fixed. That is to say, $\mathsf{SIG} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ is a stateful signature scheme if it satisfies the following.

(i) Kg is a PPT algorithm that takes as input a security parameter k and then outputs the signer's initial secret key SK_0 and public key PK. We write it $(\mathrm{SK}_0, \mathrm{PK}) \xleftarrow{\$} \mathsf{Kg}(1^k)$.

(ii) Sign is a PPT algorithm run by the signer who takes as input its stateful secret key SK_{i-1} and a message m_i and then outputs a signature Σ_i and the next stateful secret key SK_i. We write it $(\Sigma_i, \mathrm{SK}_i) \xleftarrow{\$} \mathsf{Sign}(\mathrm{SK}_{i-1}, m_i)$.

(iii) Vfy is a deterministic algorithm run by the verifier who takes as input the signer's public key PK, the signed message m_i, and the corresponding signature Σ_i and then outputs 1 if it is valid; else it outputs 0. We write it $1/0 \leftarrow \mathsf{Vfy}(\mathrm{PK}, m_i, \Sigma_i)$.

2.3. Security of Stateful Signatures in the Black-Box Model. The definition of existential unforgeability against adaptive chosen message attack (EU-CMA) for the stateful signatures is defined by the following experiment $\mathbf{Exp}^{\text{eu-cma}}_{\mathsf{SIG},\mathscr{A}}$ which is played by a EU-CMA adversary $\mathscr{A}$ and a challenger $\mathscr{B}$.

(i) $\mathscr{B}$ runs $(\mathrm{SK}_0^*, \mathrm{PK}^*) \xleftarrow{\$} \mathsf{Kg}(1^k)$ and gives PK^* to $\mathscr{A}$.

(ii) $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the following:

signing query $\mathcal{SQ}$: m_i

$\mathscr{B}$ runs $(\Sigma_i, \mathrm{SK}_i^*) \xleftarrow{\$} \mathsf{Sign}(\mathrm{SK}_{i-1}^*, m_i)$ and returns Σ_i to $\mathscr{A}$.

(iii) At some point, $\mathscr{A}$ outputs (m^*, Σ^*).

We say that $\mathscr{A}$ wins the above experiment $\mathbf{Exp}^{\text{eu-cma}}_{\mathsf{SIG},\mathscr{A}}$ if $1 \leftarrow \mathsf{Vfy}(\mathrm{PK}^*, m^*, \Sigma^*)$ and m^* was not submitted to the signing query. We denote the probability of $\mathscr{A}$ succeeded by $\mathbf{Adv}^{\text{eu-cma}}_{\mathsf{SIG},\mathscr{A}}$. We say SIG is EU-CMA secure if $\mathbf{Adv}^{\text{eu-cma}}_{\mathsf{SIG},\mathscr{A}}$ is negligible for every PPT adversary $\mathscr{A}$.

2.4. Security of Stateful Signatures in the Presence of Leakage. In the setting of the leakage-resilient cryptography, adversary $\mathscr{A}$ can obtain λ bits of leakage information with every signing

query. With the ith signing query, the adversary $\mathcal{A}$ adaptively chooses any computable leakage function $f_i: \{0,1\}^* \to \{0,1\}^\lambda$ to the leakage query and then obtains the output Λ_i of f_i which takes as input the active part SK_{i-1}^{*+} of the stateful secret key and the randomness r_i used in the signing phase. Formally, the model of existential unforgeability against adaptive chosen message and leakage attacks (EU-CMLA) is defined by the following experiment $\mathbf{Exp}_{\mathrm{SIG},\mathcal{A}}^{\text{eu-cmla}}$ which is played by a EU-CMLA adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

(i) $\mathcal{B}$ runs $(\mathrm{SK}_0^*, \mathrm{PK}^*) \xleftarrow{\$} \mathsf{Kg}(1^k)$ and gives PK^* to $\mathcal{A}$.

(ii) $\mathcal{A}$ can adaptively ask $\mathcal{B}$ for the following:

(a) signing query $\mathcal{SQ}$: m_i
$\mathcal{B}$ runs $(\Sigma_i, \mathrm{SK}_i^*) \xleftarrow{\$} \mathsf{Sign}(\mathrm{SK}_{i-1}^*, m_i, r_i)$ and returns Σ_i to $\mathcal{A}$;

(b) leakage query $\mathcal{LQ}$: f_i
$\mathcal{B}$ runs $\Lambda_i \leftarrow f_i(\mathrm{SK}_{i-1}^{*+}, r_i)$ and if $|\Lambda_i| \neq \lambda$ then it returns $\perp$; else it returns Λ_i to $\mathcal{A}$.

(iii) At some point, $\mathcal{A}$ outputs (m^*, Σ^*).

We say that $\mathcal{A}$ wins the above experiment $\mathbf{Exp}_{\mathrm{SIG},\mathcal{A}}^{\text{eu-cmla}}$ if $1 \leftarrow \mathsf{Vfy}(\mathrm{PK}, m^*, \Sigma^*)$ and m^* was not submitted to the signing query. We denote the probability of $\mathcal{A}$ succeeded by $\mathbf{Adv}_{\mathrm{SIG},\mathcal{A}}^{\text{eu-cmla}}$. We say SIG is EU-CMA secure if $\mathbf{Adv}_{\mathrm{SIG},\mathcal{A}}^{\text{eu-cmla}}$ is negligible for every PPT adversary $\mathcal{A}$.

3. Leakage-Resilient Proxy Signatures

As outlined in the Introduction, there exists three entities in a proxy signature scheme: an original signer, a (or multi) proxy signer, and a verifier. A delegator, whether it is the original signer or a proxy signer, wants to delegate its signing right, whether original signing is right (i.e., the delegator is the original signer) or proxy signing is right (i.e., the delegator is a proxy signer) to a proxy. Finally, the verifier can be convinced with the original signer's agreement on the signed message and the identities of the proxy signers from the proxy signatures.

In the multilevel proxy model, a delegation chain, (`original signer`)-(1th `proxy`)-(2th `proxy`)-$\cdots$-(jth `proxy`)-$\cdots$, consists of an original signer and j (or more) proxy signers. To identify them, we require a list $\mathcal{PK}$ of their public keys in the proxy signatures.

In the BPW transformation, the delegator will sign its proxy's public key and corresponding warrant to obtain a certificate to generate the proxy key. Therefore, to verify the validity of the delegation, it is also required that the proxy signatures contain a list $\mathcal{W}$ of the warrants and $\mathcal{C}$ of the certificates of the delegations.

3.1. Syntax. Formally, we define the stateful proxy signatures (under the BPW transformation) as follows. That is to say, $\mathsf{SIG}^* = (\mathsf{Kg}^*, \mathsf{Sign}^*, \mathsf{Vfy}^*, \langle \mathsf{Del}^*, \mathsf{PKg}^* \rangle, \mathsf{PSign}^*, \mathsf{PVfy}^*)$ is a stateful proxy signature scheme if the first three algorithms are defined as Kg, Sign, and Vfy of the scheme SIG, respectively, and the latter three algorithms satisfy the following.

(i) $\langle \mathsf{Del}^*, \mathsf{PKg}^* \rangle$ is a pair of interactive PPT delegation protocol which means that the delegator D whose stateful key is $(\mathrm{SK}_{\mathrm{D}(i-1)}, \mathrm{PK}_{\mathrm{D}})$ delegates its signing right to a proxy P who has a stateful key pair $(\mathrm{SK}_{\mathrm{P}(i'-1)}, \mathrm{PK}_{\mathrm{P}})$.

(a) Del^* is run by the delegator with input $(\mathrm{SK}_{\mathrm{D}(i-1)}, \mathrm{PK}_{\mathrm{P}}, \mathcal{PK}, \mathcal{W}, \mathcal{C}, j, W_j)$, where $\mathcal{PK}$, $\mathcal{W}$, and $\mathcal{C}$ are the lists of public keys, warrants, and delegation certificates of the previous delegators, respectively, j describes the current proxy is the jth proxy in the delegation chain ($j = 0$ means that the delegator is the original signer), and W_j is the warrant for the current delegation.

(b) PKg^* is run by the proxy with input $(\mathrm{SK}_{\mathrm{P}(i'-1)}, \mathrm{PK}_{\mathrm{P}}, \mathrm{PK}_{\mathrm{D}})$ to generate its proxy key.

As a result of this interactive algorithm, the algorithm Del^* has no local output except that the delegator's next stateful key $\mathrm{SK}_{\mathrm{D}i}$. The local output of PKg^* is the delegation information $(\mathcal{PK}', \mathcal{W}', \mathcal{C}', j, \mathrm{SK}_{\mathrm{P}(i'-1)})$, where $\mathcal{PK}'$, $\mathcal{W}'$, and $\mathcal{C}'$ are the lists of public keys, warrants, and certificates in the delegation chain extended with the public key of the proxy and warrant and certificate of the current delegation, respectively. We write it $(\mathrm{SK}_{\mathrm{D}i}, \mathcal{PK}', \mathcal{W}', \mathcal{C}', j, \mathrm{SK}_{\mathrm{P}(i'-1)}) \xleftarrow{\$} \langle \mathsf{Del}^*(\mathrm{SK}_{\mathrm{D}(i-1)}, \mathrm{PK}_{\mathrm{P}}, \mathcal{PK}, \mathcal{W}, \mathcal{C}, j, W_j), \mathsf{PKg}^*(\mathrm{SK}_{\mathrm{P}(i'-1)}, \mathrm{PK}_{\mathrm{P}}, \mathrm{PK}_{\mathrm{D}}) \rangle$.

(ii) PSign^* is a PPT algorithm run by a proxy that takes as input its delegation information $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, \mathrm{SK}_{\mathrm{P}(i'-1)})$ and a message m_i and then outputs a proxy signature $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma_i)$ on behalf of the delegator and its next stateful key $\mathrm{SK}_{\mathrm{P}i'}$. We write it $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma_i, \mathrm{SK}_{\mathrm{P}i'}) \xleftarrow{\$} \mathsf{PSign}^*(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, \mathrm{SK}_{\mathrm{P}(i'-1)}, m_i)$.

(iii) PVfy^* is a deterministic algorithm run by the verifier who takes as input $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, m_i, P\Sigma_i)$ and then outputs 1 if it is valid; else it outputs 0. We write it $1/0 \leftarrow \mathsf{PVfy}^*(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, m_i, P\Sigma_i)$.

In the real world applications, user's long-term secret key should be stored in a secure way and thus to guarantee that no information about the long-term key is leaked while the proxy key is exposed, it is better to generate a proxy key independent of the long-term key. We call such construction a full construction. There exists a simple method to the full construction from any BPW transformed proxy signature (cf. Section 5 of [15]).

(i) After obtaining the delegation information $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, \mathrm{SK}_{\mathrm{P}(i'-1)})$, the proxy first generates a fresh proxy key pair $(\mathrm{SK}'_{\mathrm{P}0}, \mathrm{PK}'_{\mathrm{P}}) \xleftarrow{\$} \mathsf{Kg}^*(1^k)$.

(ii) Compute $(\mathsf{cert}', \mathrm{SK}_{\mathrm{P}i'}) \xleftarrow{\$} \mathsf{Sign}^*(\mathrm{SK}_{\mathrm{P}(i'-1)}, 00 \parallel \mathrm{PK}'_{\mathrm{P}} \parallel 0 \parallel \mathsf{cert})$, where $\mathsf{cert} \in \mathcal{C}$ is the delegation certificate from the delegator.

(iii) The new delegation information is $(\mathcal{PK}', \mathcal{W}, \mathcal{C}', j', \mathrm{SK}'_{\mathrm{P}0})$, where $\mathrm{PK}' \in \mathcal{PK}'$ and $\mathsf{cert}' \in \mathcal{C}'$.

The concrete full construction of such proxy signature scheme and corresponding security analysis are presented in Appendices.

3.2. Implement Secure Mobile Agent from Proxy Signature Scheme. When we realize a mobile agent system construction by using a secure proxy signature scheme let the clients be the delegators and let the mobile agent be the proxy. Then the clients and the agent together run the interactive delegation protocol to delegate the client's signing right to the agent. Finally, the agent can sign some specified messages on behalf of the client. A secure proxy signature scheme implies a secure mobile agent system; similarly, a leakage-resilient proxy signature scheme means that the corresponding mobile agent system can be resilient to some bounded information leakage.

3.3. Security of the Leakage-Resilient Proxy Signatures. We put forward the security model of existential unforgeability against adaptive chosen message and leakage attacks (EU-CMLA) for the proxy signatures in the presence of leakage. It defined by the following experiment $\mathbf{Exp}^{\text{eu-cmla}}_{\text{SIG}^*,\mathscr{A}}$ which is played by a challenger $\mathscr{B}$ and a EU-CMLA adversary $\mathscr{A}$ who controls all user's secret keys except the challenging user.

(i) $\mathscr{B}$ runs $(\text{SK}_0^*, \text{PK}^*) \xleftarrow{\$} \mathsf{Kg}^*(1^k)$ and gives PK^* to $\mathscr{A}$.

(ii) $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the following:

(a) delegation to SK_{i-1}^*: PK_{D}
$\mathscr{B}$ interacts with $\mathscr{A}$ through the delegation protocol by running algorithm $\mathsf{PKg}^*(\text{SK}_{i-1}^*, \text{PK}^*, \text{PK}_{\text{D}})$. When it is finished, $\mathscr{B}$ will obtain the delegation information $(\mathscr{PK}', \mathscr{W}', \mathscr{C}', j, \text{SK}_{i-1}^*)$;

(b) delegation of SK_{i-1}^*: $(\text{PK}_{\text{P}}, W_j)$
$\mathscr{B}$ interacts with $\mathscr{A}$ through the delegation protocol to generate a proxy key to PK_{P}; $\mathscr{B}$ runs $\mathsf{Del}^*(\text{SK}_{i-1}^*, \text{PK}_{\text{P}}, \mathscr{PK}, \mathscr{W}, \mathscr{C}, j, W_j)$. When it is finished, $\mathscr{B}$ returns the transcript of the delegation to $\mathscr{A}$;

(c) self-delegation of SK_{i-1}^*: W
$\mathscr{B}$ first runs $(\text{SK}_0', \text{PK}') \xleftarrow{\$} \mathsf{Kg}^*$ and then runs the delegation protocol to generate a proxy key to the challenging user itself, $(\text{SK}_i^*, \mathscr{PK}', \mathscr{W}', \mathscr{C}', j', \text{SK}_0') \xleftarrow{\$} \langle \mathsf{Del}^*(\text{SK}_{i-1}^*, \text{PK}', \mathscr{PK}, \mathscr{W}, \mathscr{C}, j, W), \mathsf{PKg}^*(\text{SK}_0', \text{PK}', \text{PK}^*)\rangle$. When it is finished, $\mathscr{B}$ will obtain the delegation information $(\mathscr{PK}', \mathscr{W}', \mathscr{C}', j', \text{SK}_0')$ and send the transcript of the delegation to $\mathscr{A}$;

(d) ordinary signing queries of SK_{i-1}^*: m_i
$\mathscr{B}$ runs $(\Sigma_i, \text{SK}_i^*) \xleftarrow{\$} \mathsf{Sign}^*(\text{SK}_{i-1}^*, m_i)$ and returns Σ_i to $\mathscr{A}$;

(e) proxy signing queries of SK_{i-1}^*: $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, m_i)$
$\mathscr{B}$ runs $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, P\Sigma_i, \text{SK}_i^*) \xleftarrow{\$} \mathsf{PSign}^*(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, \text{SK}_{i-1}^*, m_i)$ and returns $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, P\Sigma_i)$ to $\mathscr{A}^*$;

(f) leakage queries: f_i
$\mathscr{A}$ may adaptively launches leakage query after each query to the delegation protocol, ordinary signing, or proxy signing oracle; that is, these algorithms have taken as input the secret key SK_{i-1}^*. $\mathscr{B}$ runs $\Lambda_i \leftarrow f_i(\text{SK}_{i-1}^{*+}, r_i)$ and if $|\Lambda_i| \neq \lambda$ then it returns $\perp$; else it returns Λ_i to $\mathscr{A}$.

(iii) At some point, $\mathscr{A}$ outputs a forgery which must be one of the following cases.

(1) Ordinary signature of PK^*: (m^*, Σ^*)
if $1 \leftarrow \mathsf{Vrf}^*(\text{PK}^*, m^*, \Sigma^*)$ and m^* has not been submitted to the ordinary signing queries, then output 1; else output 0.

(2) Proxy signature of PK^*: $(m^*, (\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, P\Sigma^*))$, PK^* is the last entry in $\mathscr{PK}$
if $1 \leftarrow \mathsf{PVrf}^*(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, m^*, P\Sigma^*)$ and $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, m^*)$ has not submitted to the proxy signing queries, then output 1; else output 0.

(3) Proxy signature on behalf of PK^*: $(m^*, (\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, P\Sigma^*))$, PK^* is the nth entry in $\mathscr{PK}$.
If $1 \leftarrow \mathsf{PVrf}^*(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, m^*, P\Sigma^*)$ and $\mathscr{A}$ has not queried the delegation of SK_{i-1}^* oracle on inputs $(\text{PK}_{n+1}, W_{n+1})$, that is, the $(n+1)$-th entry in the set $\mathscr{PK}$), then output 1 else output 0.

We say that $\mathscr{A}$ wins the above experiment $\mathbf{Exp}^{\text{eu-cmla}}_{\text{SIG}^*,\mathscr{A}}$ if it outputs a valid forgery. We denote the probability of $\mathscr{A}$ succeeded by $\mathbf{Adv}^{\text{eu-cmla}}_{\text{SIG}^*,\mathscr{A}}$. We say SIG^* is EU-CMLA secure if $\mathbf{Adv}^{\text{eu-cmla}}_{\text{SIG}^*,\mathscr{A}}$ is negligible for every PPT adversary $\mathscr{A}$.

Remark. In the model of EU-CMA-PKE, $\mathscr{A}$ is allowed to query a redelegation of a user's proxy key. However, we define the LRPS under the BPW transformation model (i.e., the user's proxy key is exactly its secret key), so in the model of EU-CMLA, $\mathscr{A}$ can run the redelegation by itself except that the redelegation of SK_{i-1}^* which can be obtained from the query of delegation of SK_{i-1}^* in such setting. Similarly, $\mathscr{A}$ has no need to query the proxy key exposure queries.

4. Construction of Leakage-Resilient Proxy Signatures

In this section, we present a concrete construction of the LRPS scheme SIG^* based on FKPR signature scheme which can be instantiated with any EU-CMTLA (existential unforgeability against chosen message and total leakage attacks) 3-time signature scheme $\mathsf{sig} = (\mathsf{kg}, \mathsf{sign}, \mathsf{vfy})$.

Before giving the detailed description of the SIG^*, we first introduce some notations relative to the tree-based (with depth $d \in N$) signature. We denote the all bit strings of length at most d (including the empty string ε) with $\{0,1\}^{\leq d} = \bigcup_{i=1}^{d}\{0,1\}^i \cup \varepsilon$ (size $2^{d+1} - 1$). The left and right child of an internal node (or root) $w \in \{0,1\}^{\leq d-1}$ are denoted by $w \parallel 0$ and $w \parallel 1$, respectively, and $\text{par}(w)$ denotes the node w's

parent node. Depth-first traversal algorithm can be used to traverse and label the tree. For a node $w \in \{0,1\}^{\leq d} \setminus 1^d$, we define algorithm $\mathsf{DF}(w)$ as the node traversed after w in the depth-first traversal; that is,

$$\mathsf{DF}(w) := \begin{cases} w \parallel 0, & \text{if } |w| < d \\ & (w \text{ is the root or an internal node}) \\ w' \parallel 1, & \text{if } |w| = d, \\ & \text{where } w = w' \parallel 0 \parallel 1^j \ (w \text{ is a leaf}). \end{cases} \quad (1)$$

When the depth-first algorithm traverses the binary tree, each node w is associated with a secret-public key pair $(\mathrm{sk}_w, \mathrm{pk}_w)$ by invoking the kg algorithm of the underlying signature scheme sig. The following notations will be used in the latter part of this paper. Let $w = w_1 w_2 \cdots w_t$ be a bit string with length t.

(i) $\Gamma_w := \{(\mathrm{pk}_w, \phi_w), \ldots, (\mathrm{pk}_{w_1 w_2}, \phi_{w_1 w_2}), (\mathrm{pk}_{w_1}, \phi_{w_1})\}$ is a "signature path" from w to the root; $\phi_{w'}$ is a signature of $010 \parallel \mathrm{pk}_{w'}$ with its parent's key $\mathrm{sk}_{\mathrm{par}(w')}$; that is, $\phi_{w'} \xleftarrow{\$} \mathsf{sign}(\mathrm{sk}_{\mathrm{par}(w')}, 010 \parallel \mathrm{pk}_{w'})$.

(ii) $S_w := \{\mathrm{sk}_{w_1 w_2 \cdots w_i} \mid w_{i+1} = 0\}$ is a subset of the secret keys on the path from the root ε to node w. $\mathrm{sk}_{w'} \in S_w$ if and only if the path goes to the left child $w' \parallel 0$ at the node w'. (The reason is that, in this case, the node w''s right child $w' \parallel 1$ will be traversed after node w under the depth-first traversal. Consequently, we need the secret key $\mathrm{sk}_{w'}$ of node w' to sign its right child $w' \parallel 1$'s public key $\mathrm{pk}_{w' \parallel 1}$.)

The stateful secret key of the scheme SIG^* will have the form (w, S_w, Γ_w) (i.e., using stacks S_w and Γ_w to keep track of the state, or node w). For a stack S, define the following three algorithms:

(1) $\mathsf{push}(S, a)$: putting element a on the stack S;

(2) $a \leftarrow \mathsf{pop}(S)$: removing the topmost element from the stack S and assigning it to a;

(3) $\mathsf{trash}(S)$: removing the topmost element from the stack S.

4.1. Construction. To avoid trivial attacks against this scheme, we use the idea of Boldyreva et al. [13], attach a 3-bit string as the prefix of the text that will be signed, that is, 111‖(text which will be to compute ordinary signatures), 010‖(text which will be to compute signature paths), 100‖(text which will be to compute delegation certificates), and 101‖(text which will be to compute proxy signatures), respectively. The LRPS scheme SIG^* is constructed as follows.

(i) $\mathsf{Kg}^*(1^k)$:

$(\mathrm{sk}_\varepsilon, \mathrm{pk}_\varepsilon) \xleftarrow{\$} \mathsf{Kg}(1^k), S_\varepsilon := \mathrm{sk}_\varepsilon, \Gamma_\varepsilon := \emptyset, \mathrm{SK}_\varepsilon := (w_\varepsilon, S_\varepsilon, \Gamma_\varepsilon), \mathrm{PK} := \mathrm{pk}_\varepsilon$; return $(\mathrm{SK}_\varepsilon, \mathrm{PK})$.

(ii) $\mathsf{Sign}^*(\mathrm{SK}_w, m)$: (to ease exposition, the signing process of the root ε (i.e., $\sigma \xleftarrow{\$} \mathsf{sign}(\mathrm{sk}_\varepsilon, 111 \parallel m)$) is not contained in this formalizing description)

parse $\mathrm{SK}_w := (w, S_w, \Gamma_w)$; if $w = 1^d$ return $\perp$; $\widehat{w} \leftarrow \mathsf{DF}(w)$, $(\mathrm{sk}_{\widehat{w}}, \mathrm{pk}_{\widehat{w}}) \xleftarrow{\$} \mathsf{Kg}(1^k)$

$\sigma \xleftarrow{\$} \mathsf{sign}(\mathrm{sk}_{\widehat{w}}, 111 \parallel m)$; $\mathrm{sk}_{\mathrm{par}(\widehat{w})} \leftarrow \mathsf{pop}(S_w)$; $\phi_{\widehat{w}} \xleftarrow{\$} \mathsf{sign}(\mathrm{sk}_{\mathrm{par}(\widehat{w})}, 010 \parallel \mathrm{pk}_{\widehat{w}})$

$$\begin{aligned} &\text{if } \widehat{w}_{|\widehat{w}|} = 0, \quad S_w \longleftarrow \mathsf{push}\left(S_w, \mathrm{sk}_{\mathrm{par}(\widehat{w})}\right) \\ &\quad \text{if } |\widehat{w}| < d, \quad S_{\widehat{w}} \longleftarrow \mathsf{push}\left(S_w, \mathrm{sk}_{\widehat{w}}\right) \\ &\qquad \text{if } |w| = d, \qquad w = w' 01^j, \\ &\quad \text{for } i = 1, \ldots, j+1, \text{ do } \mathsf{trash}\left(\Gamma_w\right) \end{aligned} \quad (2)$$

$\Gamma_{\widehat{w}} \leftarrow \mathsf{push}(\Gamma_w, (\mathrm{pk}_{\widehat{w}}, \phi_{\widehat{w}}))$; $\Sigma := (\sigma, \Gamma_{\widehat{w}})$; $\mathrm{SK}_{\widehat{w}} := (\widehat{w}, S_{\widehat{w}}, \Gamma_{\widehat{w}})$; return $(\Sigma, \mathrm{SK}_{\widehat{w}})$.

(iii) $\mathsf{Vfy}^*(\mathrm{PK}, m, \Sigma)$:

parse $\Sigma := (\sigma, \Gamma_{\widehat{w}_1 \widehat{w}_2 \cdots \widehat{w}_{|\widehat{w}|}})$, $\mathrm{pk}_\varepsilon := \mathrm{PK}$; for $i = 1, \ldots, |\widehat{w}|$ do

if $0 \leftarrow \mathsf{vfy}(\mathrm{pk}_{\widehat{w}_1 \cdots \widehat{w}_{i-1}}, 010 \parallel \mathrm{pk}_{\widehat{w}_1 \cdots \widehat{w}_i}, \phi_{\widehat{w}_1 \cdots \widehat{w}_i})$ return 0;

else return $\mathsf{vfy}(\mathrm{pk}_{\widehat{w}_1 \widehat{w}_2 \cdots \widehat{w}_{|\widehat{w}|}}, 111 \parallel m, \sigma)$.

(iv) $\mathsf{Del}^*(\mathrm{SK}_{\mathrm{D}(i-1)}, \mathrm{PK}_\mathrm{P}, \mathscr{PK}, \mathscr{W}, \mathscr{C}, j, W_j)$:

D runs $(\mathsf{cert}_j, \mathrm{SK}_{\mathrm{D}i}) \xleftarrow{\$} \mathsf{Sign}^*(\mathrm{SK}_{\mathrm{D}(i-1)}, 100 \parallel \mathrm{PK}_\mathrm{P} \parallel j \parallel W_j)$ and

then sends $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, W_j, \mathsf{cert}_j)$ to P.

(v) $\mathsf{PKg}^*(\mathrm{SK}_{\mathrm{P}(i'-1)}, \mathrm{PK}_\mathrm{P}, \mathrm{PK}_\mathrm{D})$:

P first checks the validity of the delegation certificates, for $k = 1, \ldots, j$ does

if $0 \leftarrow \mathsf{Vfy}^*(\mathrm{PK}_{k-1}, 100 \parallel \mathrm{PK}_k \parallel k \parallel W_k, \mathsf{cert}_k)$, it returns $\perp$ and rejects this delegation;

otherwise, run $\mathscr{PK} \leftarrow \mathsf{push}(\mathscr{PK}, \mathrm{PK}_\mathrm{P})$, $\mathscr{W} \leftarrow \mathsf{push}(\mathscr{W}, W_j)$, $\mathscr{C} \leftarrow \mathsf{push}(\mathscr{C}, \mathsf{cert}_j)$;

finally, set the delegation information as $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, \mathrm{SK}_{\mathrm{P}(i'-1)})$.

If someone, whose key pair is $(\mathrm{SK}_{\mathrm{SD}(i-1)}, \mathrm{PK}_{\mathrm{SD}})$, wants to designate itself as a proxy it runs $(\mathrm{SK}'_{\mathrm{P}0}, \mathrm{PK}'_\mathrm{P}) \xleftarrow{\$} \mathsf{Kg}^*(1^k)$ to generate a fresh key pair as the proxy key and creates a certificate $(\mathsf{cert}', \mathrm{SK}_{\mathrm{SD}i}) \xleftarrow{\$} \mathsf{Sign}^*(\mathrm{SK}_{\mathrm{SD}(i-1)}, 100 \parallel \mathrm{PK}'_\mathrm{P} \parallel 0 \parallel W')$, then does

$$\begin{aligned} &\mathscr{PK} \longleftarrow \mathsf{push}\left(\mathscr{PK}, \mathrm{PK}'_\mathrm{P}\right), \\ &\mathscr{W} \longleftarrow \mathsf{push}\left(\mathscr{W}, W'\right), \\ &\mathscr{C} \longleftarrow \mathsf{push}\left(\mathscr{C}, \mathsf{cert}'\right); \end{aligned} \quad (3)$$

finally, it sets the delegation information as $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, \mathrm{SK}'_{\mathrm{P}0})$.

(vi) $\mathsf{PSign}^*(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, \mathrm{SK}_{\mathrm{P}(i-1)}, m)$:

$(\Sigma, \mathrm{SK}_{\mathrm{P}i}) \xleftarrow{\$} \mathsf{Sign}^*(\mathrm{SK}_{\mathrm{P}(i-1)}, 101 \parallel m)$ and output the proxy signature $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma := \Sigma)$.

(vii) $\mathsf{PVfy}^*(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, m, P\Sigma)$:

V first checks the validity of the delegation certificates, for $k = 1, \ldots, j$ does

if $0 \leftarrow \mathsf{Vfy}^*(\mathrm{PK}_{k-1}, 100 \parallel \mathrm{PK}_k \parallel k \parallel W_k, \mathsf{cert}_k)$ returns 0;

else it returns $\mathsf{Vfy}^*(\mathrm{PK}_j, 101 \parallel m, P\Sigma)$.

Upper Bound of the Number of the Messages Can Be Signed. For a fixed signing key, in both of the schemes FKPR and SIG^*, the upper bound of the number of the message that can be signed is $q = 2^{d+1} - 2$. We can see that, from the above construction, each internal node is used only one time to the signing algorithm. However, the key (with respect to the scheme sig) of any leaf can be signed three times. Hence, the upper bound of the number of the message can be signed and could be increased to $2^{d+2} - 4$ that is double the number of the previous upper bound, as well as the FKPR scheme.

We should stress here that there is a disadvantage to our scheme which is based on tree-based signature compared to that constructed based on aggregate signature [13, 15]; that is, in those schemes, the verification of the delegation certificates can be executed at a time due to the property of aggregability of the aggregate signatures [41].

4.2. Security. We now analyze the security of the proposed LRPS scheme.

Theorem 1. *If the FKPR scheme (denoted by* SIG*) is EU-CMLA secure, then the proxy signature scheme* SIG^* *also is EU-CMLA secure.*

Our proof line is similar to that of Boldyreva et al.'s [13]. If there exists a EU-CMLA adversary and $\mathcal{A}$ can break the security of the scheme SIG^*, then we can construct a challenger $\mathcal{B}$ to break the security of the FKPR scheme SIG.

(i) Initially, $\mathcal{B}$ will be given a challenging public key PK' and can adaptively make signing query ($\mathcal{SQ}$) and leakage query ($\mathcal{LQ}$) in the experiment $\mathbf{Exp}^{\text{eu-cmla}}_{\mathsf{SIG},\mathcal{B}}$. $\mathcal{B}$ first sets $\mathrm{PK}^* := \mathrm{PK}'$ as the challenging public key of the experiment $\mathbf{Exp}^{\text{eu-cmla}}_{\mathsf{SIG}^*,\mathcal{A}}$ and sends it to $\mathcal{A}$. Then it plays the experiment with $\mathcal{A}$.

(ii) $\mathcal{A}$ may adaptively ask $\mathcal{B}$ for the following.

(a) Delegation to SK^*_{i-1}: PK_{D}

$\mathcal{B}$ interacts with $\mathcal{A}$ through the delegation protocol by running $\mathsf{PKg}^*(*, \mathrm{PK}^*, \mathrm{PK}_{\mathrm{D}})$. When it is finished, $\mathcal{B}$ will obtain the delegation information $(\mathcal{PK}', \mathcal{W}', \mathcal{C}', j, *)$. $\mathcal{B}$ can run the PKg^* algorithm even if it has no idea about the SK^*_{i-1}, because SK^*_{i-1} will be set as the proxy key of the challenging user, so upon completion, $\mathcal{B}$ does not know the corresponding proxy key.

(b) Delegation from SK^*_{i-1}: $(\mathrm{PK}_{\mathrm{P}}, W_j)$

$\mathcal{B}$ interacts with $\mathcal{A}$ through the delegation protocol to generate a proxy key to PK_{P}. $\mathcal{B}$ makes the signing query $\mathcal{SQ}$ with input $00 \parallel \mathrm{PK}_{\mathrm{P}} \parallel j \parallel W_j$; then it will be returned Σ. After the delegation protocol is finished, $\mathcal{A}$ will obtain the delegation information $(\mathcal{PK}', \mathcal{W}', \mathcal{C}', j, *)$, where $\mathrm{PK}_{\mathrm{P}} \in \mathcal{PK}'$, $W_j \in \mathcal{W}'$, and $\mathsf{cert}_j := \Sigma \in \mathcal{C}'$.

(c) Self-delegation of SK^*_{i-1}: W

$\mathcal{B}$ runs the delegation protocol to generate a proxy key of PK^* to itself. $\mathcal{B}$ first runs $(\mathrm{SK}'_0, \mathrm{PK}') \xleftarrow{\$} \mathsf{Kg}^*$ and then makes the signing query $\mathcal{SQ}$ with input $00 \parallel \mathrm{PK}' \parallel 0 \parallel W$; then it will be returned to Σ. Finally, $\mathcal{B}$ will return the delegation information $(\mathcal{PK}', \mathcal{W}', \mathcal{C}', 0, \mathrm{SK}'_0)$ and sends the delegation transcripts to $\mathcal{A}$, where $\mathrm{PK}' \in \mathcal{PK}', W \in \mathcal{W}'$, and $\mathsf{cert}' := \Sigma \in \mathcal{C}'$.

(d) Ordinary signing queries of SK^*_{i-1}: m_i

$\mathcal{B}$ makes the signing query $\mathcal{SQ}$ with input $11 \parallel m_i$; then it will be returned to signature Σ. Finally, $\mathcal{B}$ returns Σ to $\mathcal{A}$.

(e) Proxy signing queries of SK^*_{i-1}: $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, m_i)$

$\mathcal{B}$ makes the signing query $\mathcal{SQ}$ with input $01 \parallel m_i$; then it will be returned to signature Σ. Finally, $\mathcal{B}$ returns $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma := \Sigma)$ to $\mathcal{A}$.

(f) Leakage queries: f_i

$\mathcal{A}$ may make query f_i for the leakage information after each delegation protocol, ordinary signing, or proxy signing query. To answer it, $\mathcal{B}$ makes the same query to $\mathcal{LQ}$; it will be returned as a valid leakage information Λ_i or $\perp$ if f_i is illegal. Finally, $\mathcal{B}$ returns it to $\mathcal{A}$.

Remark. In the construction of scheme SIG^*, except for the Sign^* algorithm, there are also two algorithms using the signing or proxy signing key, the Del^* and PSign^*. Actually, however, they are also a signing algorithm just with different input of text, so the leakage information answered by $\mathcal{B}$ (from $\mathcal{LQ}$) is indistinguishable to what $\mathcal{A}$ obtains in the real interaction in the experiment $\mathbf{Exp}^{\text{eu-cmla}}_{\mathsf{SIG}^*,\mathcal{A}}$.

(iii) Finally, according to the assumption, $\mathcal{A}$ outputs a forgery for the challenging public key PK^* with respect to scheme SIG^*. It must be one of the following cases. We now show the challenger $\mathcal{B}$ how to translate $\mathcal{A}$'s forgery as a forgery with respect to the FKPR scheme SIG.

(1) Ordinary signature of PK^*: (m^*, Σ^*)

If $\mathcal{A}$ outputs an ordinary signature (m^*, Σ^*) of PK^*, then $\mathcal{B}$ outputs $(11 \parallel m^*, \Sigma^*)$.

(2) Proxy signature of PK^*: $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma^*))$, PK^* is the last entry in $\mathcal{PK}$.

If $\mathcal{A}$ outputs a proxy signature $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma^*))$ of PK^*, $\mathcal{B}$ outputs $(01 \parallel m^*, \Sigma^*)$.

(3) Proxy signature on behalf of PK^*: $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma^*))$, PK^* is the nth entry in the list $\mathcal{PK}$.

If $\mathcal{A}$ outputs a proxy signature $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma^*))$ on behalf of PK^*, then $\mathcal{B}$ outputs $(00 \parallel PK_{n+1} \parallel n+1 \parallel W_n, \mathsf{cert}_{n+1})$.

Analysis of $\mathcal{B}$. It is clear that the view of $\mathcal{A}$ which is answered by $\mathcal{B}$ in the above experiment is identical to what $\mathcal{A}$ obtains in the real interaction in the experiment $\mathbf{Exp}^{\text{eu-cmla}}_{\mathsf{SIG}^*,\mathcal{A}}$. We now show that any valid output of the adversary $\mathcal{A}$ can be translated to a valid forgery with respect to the FKPR scheme SIG.

(1) If $\mathcal{A}$ outputs an ordinary signature (m^*, Σ^*), $1 \leftarrow \mathsf{Vrf}^*(PK^*, m^*, \Sigma^*)$, and m^* has not been submitted to the ordinary signing queries, so $\mathcal{B}$ does not make the signing query $\mathcal{SQ}$ with input $11 \parallel m^*$. Therefore, $(11 \parallel m^*, \Sigma^*)$ is a valid forgery with respect to the scheme SIG.

(2) If $\mathcal{A}$ outputs a proxy signature $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma^*))$, $1 \leftarrow \mathsf{PVrf}^*(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, m^*, P\Sigma^*)$, and $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, m^*)$ has not submitted to the proxy signing queries, so $\mathcal{B}$ does not make the signing query $\mathcal{SQ}$ with input $01 \parallel m^*$. Therefore, $(01 \parallel m^*, P\Sigma^*)$ is a valid forgery with respect to the scheme SIG.

(3) If $\mathcal{A}$ outputs a proxy signature on behalf of PK^*: $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, P\Sigma^*))$, where PK^* is the nth entry in $\mathcal{PK}$, $1 \leftarrow \mathsf{PVrf}^*(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, m^*, P\Sigma^*)$ and $\mathcal{A}$ does not make the query of delegation from SK^*_{i-1} with input (PK_{n+1}, W_{n+1}) ($(n+1)$th entry in $\mathcal{PK}$), so $\mathcal{B}$ does not make the signing query $\mathcal{SQ}$ with input $00 \parallel PK_{n+1} \parallel n+1 \parallel W_n$. Therefore, $(00 \parallel PK_{n+1} \parallel n+1 \parallel W_n, \mathsf{cert}_{n+1})$ is a valid forgery with respect to the scheme SIG.

From the above analysis, we can see that the challenger $\mathcal{B}$'s output of forgery is contradictory to the security of the FKPR scheme SIG (cf. Theorem 1 of [37]) and thus proves the security of the LRPS scheme SIG^*.

5. Conclusion

In this paper, we design a leakage-resilient proxy signature scheme, the LRPS. To model the security of such schemes, we adapt the existing models of the proxy signature schemes which are proposed by Schuldt et al. (in PKC 2008) [15] and Boldyreva et al. (in Jour. Crypto. 2012) [13] to the leakage-resilient cryptography setting and give an extended model, EU-CMLA, for the LRPS schemes. Furthermore, we present a concrete construction based on Faust et al.'s (in TCC 2010) [37] LR signature scheme. This construction is provably secure under the given security model.

Appendices

Now we show that their proposed proxy signature scheme SIG^* in Section 4 which is based on the BPW transformation can be used to produce a secure full construction (denoted by SIG^{**}) of the proxy signature scheme.

A. Construction

As said before, to guarantee that no information about the user's long-term secret key is leaked if its proxy keys are exposed, we had better let a proxy generate fresh and independent keys (PK, SK) in a delegation, create a certificate for PK, and keep the SK as the proxy secret key; to record the proxy public keys of the proxies maintain a separate list $\mathcal{FK}$ to store them. The construction of the scheme $\mathsf{SIG}^{**} = (\mathsf{Kg}^{**}, \mathsf{Sign}^{**}, \mathsf{Vfy}^{**}, \langle \mathsf{Del}^{**}, \mathsf{PKg}^{**} \rangle, \mathsf{PSign}^{**}, \mathsf{PVfy}^{**})$ is as follows, where the algorithms $\mathsf{Kg}^{**}, \mathsf{Sign}^{**}, \mathsf{Vfy}^{**}$ are the same as the algorithms $\mathsf{Kg}^*, \mathsf{Sign}^*, \mathsf{Vfy}^*$ of the scheme SIG^*, respectively. Here we should stress that the following construction is based on Schuldt et al.'s [15] idea, while their scheme is based on sequential aggregate signature, but ours is based on tree-based signature and we focus on the realization of the leakage-resilient proxy signature.

In the scheme SIG^*, the proxy's proxy key is in fact exactly its long-term secret key and hence it delegates its own signing right or proxy's signing right to the next proxy, it takes as input its secret key to run the delegation algorithm Del^*. However, when we consider the full construction of the proxy signature scheme, proxy's secret key and proxy's key are different and independent, and thus when it delegates its own signing right to a proxy it takes as input its secret key; when it delegates its proxy signing right to the next proxy, then it takes as input the proxy key. To uniformly describe these two cases, we use sk to denote the input to the Del^{**} algorithm run by the delegator in the scheme SIG^{**}. For ease of description, here we describe the stateful signing algorithm Sign^{**} as a nonstateful formalization.

(i) $\mathsf{Del}^{**}(\mathrm{sk}, PK_P, \mathcal{PK}, \mathcal{FK}, \mathcal{W}, \mathcal{C}, W)$: it is divided into the following two cases depending on $(\mathcal{PK}, \mathcal{W})$

(a) If $\mathcal{PK}$ and $\mathcal{W}$ are empty (i.e., sk is an long-term secret key), the delegator constructs lists $\mathcal{PK} = \{PK_D, PK_P\}$, $\mathcal{FK} = \emptyset$, and $\mathcal{W} = \{W\}$. Then compute $\mathsf{cert} \xleftarrow{\$} \mathsf{Sign}^{**}(\mathrm{sk}, 100 \parallel \mathcal{PK} \parallel \mathcal{FK} \parallel \mathcal{W})$ and send the delegation information $(\mathcal{PK}, \mathcal{FK}, \mathcal{W}, \mathsf{cert})$ to the proxy.

(b) If $\mathcal{PK}$ and $\mathcal{W}$ are not empty (i.e., sk is a proxy key), the delegator constructs lists $\mathcal{PK} \leftarrow \mathsf{push}(\mathcal{PK}, PK_P)$ and $\mathcal{W} \leftarrow \mathsf{push}(\mathcal{W}, W)$. Then compute $\mathsf{cert} \xleftarrow{\$} \mathsf{Sign}^{**}(\mathrm{sk}, 100 \parallel \mathcal{PK} \parallel \mathcal{FK} \parallel \mathcal{W})$ and send the delegation information $(\mathcal{PK}, \mathcal{FK}, \mathcal{W}, \mathcal{C}, \mathsf{cert})$ to the proxy.

(ii) $\mathsf{PKg}^{**}(\mathrm{SK_P}, \mathrm{PK_P}, \mathrm{PK_D})$:

the proxy first checks the validity of the delegation certificates for $k = 1, \ldots, |\mathscr{C}|$ does: if $0 \leftarrow \mathsf{Vfy}^{**}(\mathrm{PK}_{k-1}, 100 \parallel \mathscr{PK} \parallel \mathscr{FK} \parallel \mathscr{W}, \mathsf{cert}_k)$, it returns $\perp$ and rejects this delegation, where cert_k means the kth entry in the list $\mathscr{C}$. Otherwise, first generate a fresh proxy key pair $(\mathrm{PK}'_{\mathrm{P}}, \mathrm{SK}'_{\mathrm{P}}) \leftarrow \mathsf{Kg}^{**}(1^k)$ and run $\mathscr{FK} \leftarrow \mathsf{push}(\mathscr{FK}, \mathrm{PK}'_{\mathrm{P}})$. Then compute $\mathsf{cert} \xleftarrow{\$} \mathsf{Sign}^{**}(\mathrm{SK_P}, 100 \parallel \mathscr{PK} \parallel \mathscr{FK} \parallel \mathscr{W})$. Finally, run $\mathscr{PK} \leftarrow \mathsf{push}(\mathscr{PK}, \mathrm{PK_P}), \mathscr{W} \leftarrow \mathsf{push}(\mathscr{W}, W), \mathscr{C} \leftarrow \mathsf{push}(\mathscr{C}, \mathsf{cert})$; set $\mathrm{PSK} = (\mathscr{FK}, \mathsf{cert}, \mathrm{SK}'_{\mathrm{P}})$ and output the delegation information $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, \mathrm{PSK})$.

(iii) $\mathsf{PSign}^{**}(\mathscr{PK}, \mathscr{W}, \mathscr{C}, \mathrm{PSK}, m)$:

$\Sigma \xleftarrow{\$} \mathsf{Sign}^{**}(\mathrm{SK}'_{\mathrm{P}}, 101 \parallel m)$, output the proxy signature $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, P\Sigma := \Sigma)$.

(iv) $\mathsf{PVfy}^{**}(\mathscr{PK}, \mathscr{FK}, \mathscr{W}, \mathscr{C}, m, P\Sigma)$:

V first checks the validity of the delegation certificates, for $k = 1, \ldots, |\mathscr{C}|$ does $\mathsf{Vfy}^{**}(\mathrm{PK}_{k-1}, 100 \parallel \mathscr{PK} \parallel \mathscr{FK} \parallel \mathscr{W}, \mathsf{cert}_k)$ or $\mathsf{Vfy}^{**}(\mathrm{PK}'_{k-1}, 100 \parallel \mathscr{PK} \parallel \mathscr{FK} \parallel \mathscr{W}, \mathsf{cert}_k)$ dependent on the current certificate generated by Del^{**} or PKg^{**}, respectively. If all the verifications pass then return $\mathsf{Vfy}^{**}(\mathrm{PK}'_{\mathrm{P}}, 101 \parallel m, P\Sigma)$.

B. Security

We now analyze the security of the scheme SIG^{**}. This proof is roughly analogous to the proof of scheme SIG^{*}. However, because the proxy key is independent of the long-term secret key, we have to permit more queries to the adversary, such as a redelegation of a user's proxy key. Here we adapt Schuldt et al.'s [15] security model, EU-CMA-PKE which is the strongest notion for the proxy signature schemes (cf. Section 4 of [15] for detailed description), to the leakage-resilient cryptography setting, EU-CMLA-PKE. In the presence of leakage, we should care about what secret can be taken as input to the leakage function: long-term secret key, proxy key, or both? Our answer is both.

The detailed analysis is as follows.

Theorem B.1. *The proxy signature scheme* SIG^{**} *is EU-CMLA-PKE secure based on the security of the leakage-resilient FKPR signature scheme* SIG.

We show that if there exists a EU-CMLA-PKE adversary $\mathscr{A}$ which can break the security of the scheme SIG^{**}, then it can be used to construct a challenger $\mathscr{B}$ to break the security of the FKPR scheme SIG.

(I) Initially, $\mathscr{B}$ will be given a challenging public key PK' and can adaptively make signing query ($\mathcal{SQ}$) and leakage query ($\mathcal{LQ}$) in the experiment $\mathbf{Exp}^{\text{eu-cmla}}_{\mathrm{SIG}^*,\mathscr{B}}$. $\mathscr{B}$ first chooses a random $c \leftarrow \{0, 1\}$. If $c = 0$, $\mathscr{B}$ sets $\mathrm{PK}^* := \mathrm{PK}'$ and $\mathrm{SK}^* := \emptyset$. Otherwise, $\mathscr{B}$ generates a fresh key pair $(\mathrm{PK}^*, \mathrm{SK}^*) \leftarrow \mathsf{Kg}^{**}$ and chooses random $i^* \leftarrow \{1, \ldots, q_d\}$ (where q_d is the number that $\mathscr{A}$ queries to the delegation oracle; $\mathscr{B}$ will use PK' instead of a fresh key in the i^*th delegation query by $\mathscr{A}$). For both cases, $\mathscr{B}$ sends PK^* to $\mathscr{A}$ as the challenging public key of the experiment $\mathbf{Exp}^{\text{eu-cmla-pke}}_{\mathrm{SIG}^*,\mathscr{A}}$. Then it plays the experiment with $\mathscr{A}$.

(II) $\mathscr{A}$ may adaptively ask $\mathscr{B}$ for the following. When the queries by $\mathscr{A}$ need signing invocation of SK' corresponding to PK', $\mathscr{B}$ queries its own singing oracle $\mathcal{SQ}$, and we omit this implicit description in the following proof. In addition, $\mathscr{B}$ will maintain a set of lists $\mathrm{PskList}(*, *)$ which contains all proxy keys generated by $\mathscr{B}$ for the delegation chain with the public keys $\mathscr{PK}$ and warrants $\mathscr{W}$.

(i) Delegation to SK^*: $(\mathscr{PK}, \mathscr{FK}, \mathscr{W}, \mathscr{C})$

if $c = 0$, or $c = 1$ and this is not the i^*th delegation query, then $\mathscr{B}$ first runs $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Kg}^{**}(1^k)$, $\mathscr{FK} \leftarrow \mathsf{push}(\mathscr{FK}, \mathrm{PK})$ and set $\mathrm{SK}_{\mathrm{prx}} = \mathrm{SK}$. If $c = 1$ and this is the i^*th delegation query, $\mathscr{B}$ runs $\mathscr{FK} \leftarrow \mathsf{push}(\mathscr{FK}, \mathrm{PK}^*)$ and set $\mathrm{SK}_{\mathrm{prx}} = \emptyset$. Then $\mathscr{B}$ computes $\mathsf{cert} \leftarrow \mathsf{Sign}^{**}(\mathrm{SK}_{\mathrm{prx}}, 100 \parallel \mathscr{PK} \parallel \mathscr{FK} \parallel \mathscr{W})$. Finally, store $\mathrm{PSK} = (\mathscr{FK}, \mathsf{cert}, \mathrm{SK}_{\mathrm{prx}})$ in $\mathrm{PskList}(\mathscr{PK}, \mathscr{W})$.

(ii) Delegation from SK^*: this query can be divided into the following three cases.

(a) Delegation of SK^*: $(\mathrm{PK_P}, W)$

$\mathscr{B}$ sets $\mathscr{PK} = \{\mathrm{PK}^*, \mathrm{PK_P}\}$, $\mathscr{FK} = \emptyset$, and $\mathscr{W} = \{W\}$. Then compute $\mathsf{cert} \leftarrow \mathsf{Sign}^{**}(\mathrm{SK}^*, 100 \parallel \mathscr{PK} \parallel \mathscr{FK} \parallel \mathscr{W})$ and set $\mathscr{C} = \{\mathsf{cert}\}$. Finally return the delegation information $(\mathscr{PK}, \mathscr{FK}, \mathscr{W}, \mathscr{C})$ to $\mathscr{A}$.

(b) Redelegation of PSK: $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, \mathrm{PK_P}, W)$

$\mathscr{B}$ retrieves the jth proxy key $\mathrm{PskList}(\mathscr{PK}, \mathscr{W})$ and parses it as $(\mathscr{FK}, \mathsf{cert}, \mathrm{SK}_{\mathrm{prx}})$. Then run $\mathscr{PK} \leftarrow \mathsf{push}(\mathscr{PK}, \mathrm{PK_P}), \mathscr{W} \leftarrow \mathsf{push}(\mathscr{W}, W)$, compute $\mathsf{cert} \leftarrow \mathsf{Sign}^{**}(\mathrm{SK}_{\mathrm{prx}}, 100 \parallel \mathscr{PK} \parallel \mathscr{FK} \parallel \mathscr{W})$, and set$\mathscr{C} \leftarrow \mathsf{push}(\mathscr{C}, \mathsf{cert})$. Finally return the delegation information $(\mathscr{PK}, \mathscr{FK}, \mathscr{W}, \mathscr{C})$ to $\mathscr{A}$.

(c) Self-delegation of SK^*: $(\mathscr{PK}, \mathscr{W}, \mathscr{C}, j, W)$

(1) if $\mathscr{PK}$ and $\mathscr{W}$ are empty (i.e., self-delegation of SK^*), $\mathscr{B}$ constructs $\mathscr{PK} = \{\mathrm{PK}^*, \mathrm{PK}^*\}$, $\mathscr{FK} = \emptyset$, and $\mathscr{W} = \{W\}$ and sets $\mathrm{SK}_{\mathrm{sel}} = \mathrm{SK}^*$ and $\mathsf{cert}_{\mathrm{sel}} = \emptyset$.

(2) If $\mathscr{PK}$ and $\mathscr{W}$ (i.e., delegation of PSK), $\mathscr{B}$ retrieves the jth proxy key in $\mathrm{PskList}(\mathscr{PK}, \mathscr{W})$ and parses it as $(\mathscr{FK}, \mathsf{cert}, \mathrm{SK}_{\mathrm{prx}})$. Then compute $\mathscr{PK} \leftarrow \mathsf{push}(\mathscr{PK}, \mathrm{PK}^*), \mathscr{W} \leftarrow \mathsf{push}(\mathscr{W}, W)$, and set $\mathrm{SK}_{\mathrm{sel}} = \mathrm{SK}_{\mathrm{prx}}$ and $\mathsf{cert}_{\mathrm{sel}} = \mathsf{cert}$.

$\mathscr{B}$ then computes $\mathsf{cert} \leftarrow \mathsf{Sign}^{**}(\mathrm{SK}_{\mathrm{sel}}, 100 \parallel \mathscr{PK} \parallel \mathscr{FK} \parallel \mathscr{W})$. If $c = 0$ or $c = 1$ and this not the i^*th delegation query, $\mathscr{B}$ first runs $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Kg}^{**}(1^k)$, and construct $\mathscr{FK} \leftarrow \mathsf{push}(\mathscr{FK}, \mathrm{PK})$. Otherwise, $\mathscr{B}$ constructs $\mathscr{FK} \leftarrow \mathsf{push}(\mathscr{FK}, \mathrm{PK}^*)$, and set $\mathrm{SK} = \emptyset$. Finally, $\mathscr{B}$ computes $\mathsf{cert} \leftarrow \mathsf{Sign}^{**}(\mathrm{SK}_{\mathrm{sel}}, 100 \parallel$

$\mathcal{PK} \parallel \mathcal{FK} \parallel \mathcal{W}$) and $\mathcal{C} \leftarrow \mathsf{push}(\mathcal{C}, \mathsf{cert})$, and then store the proxy key PSK = $(\mathcal{FK}, \mathsf{cert}, \mathrm{SK})$ in PskList$(\mathcal{PK}, \mathcal{W})$ and send the transcript $(\mathcal{PK}, \mathcal{FK}, \mathcal{W}, \mathcal{C})$ to $\mathcal{A}$.

(iii) Ordinary signing queries of SK^*: m_i

$\mathcal{B}$ returns $\mathsf{Sign}^{**}(\mathrm{SK}^*, 111 \parallel m)$.

(iv) Proxy signing queries of SK^*: $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, j, m_i)$

$\mathcal{B}$ retrieves the jth proxy key in PskList$(\mathcal{PK}, \mathcal{W})$ and parses it as $(\mathcal{FK}, \mathsf{cert}, \mathrm{SK}_{\mathrm{prx}})$. Then compute $P\Sigma \leftarrow \mathsf{PSign}^{**}(\mathrm{SK}_{\mathrm{prx}}, 101 \parallel m_i)$ and return $(\mathcal{PK}, \mathcal{W}, \mathcal{C}, (\mathcal{FK}, P\Sigma))$ to $\mathcal{A}$.

(v) Proxy key exposure queries: $(\mathcal{PK}, \mathcal{W}, j)$

$\mathcal{B}$ retrieves the jth proxy key in PskList$(\mathcal{PK}, \mathcal{W})$ and parses it as $(\mathcal{FK}, \mathsf{cert}, \mathrm{SK}_{\mathrm{prx}})$. If $\mathrm{SK}_{\mathrm{prx}} = \emptyset$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns $(\mathcal{FK}, \mathsf{cert}, \mathrm{SK}_{\mathrm{prx}})$ to $\mathcal{A}$.

(vi) Leakage queries: f_i:

$\mathcal{A}$ makes query f_i for the leakage information about the secret key sk (randomness is also included here) after each delegation protocol, ordinary signing, or proxy signing query. If the used secret key is chosen by $\mathcal{B}$, then $\mathcal{B}$ returns $\Lambda_i = f_i(\mathrm{sk})$. Otherwise, $\mathcal{B}$ makes the same query to its own leakage oracle $\mathcal{LQ}$, it will be returned as valid leakage information Λ_i or $\perp$ if f_i is illegal. Finally, $\mathcal{B}$ returns it to $\mathcal{A}$.

Remark. The secret state for $\mathcal{A}$ can be divided into two kinds, the first one is that chosen by $\mathcal{B}$ in the experiment, and the second one is that unknown to $\mathcal{B}$, that is, SK' and the randomness used in the singing oracle $\mathcal{SQ}$. For the first one, $\mathcal{B}$ can directly answer $\mathcal{A}$ by itself. For the second one, similar to the proof in Theorem 1, $\mathcal{B}$ can make the same query to its leakage oracle $\mathcal{LQ}$.

(III) Finally, according to the assumption, $\mathcal{A}$ outputs a forgery for the challenging public key PK^* (with respect to the scheme SIG^{**}). It must be one of the following cases:

(1) ordinary signature (m^*, Σ^*);

(2) proxy signature $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, (\mathcal{FK}, P\Sigma^*)))$, where the last key in $\mathcal{FK}$ was not generated by $\mathcal{B}$;

(3) proxy signature $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, (\mathcal{FK}, P\Sigma^*)))$, where the $(i^* - 1)$th key in $\mathcal{FK}$ was not generated by $\mathcal{B}$;

(4) proxy signature $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, (\mathcal{FK}, P\Sigma^*)))$, where the last key in $\mathcal{FK}$ was generated by $\mathcal{B}$;

(5) proxy signature $(m^*, (\mathcal{PK}, \mathcal{W}, \mathcal{C}, (\mathcal{FK}, P\Sigma^*)))$, where the $(i^* - 1)$th key in $\mathcal{FK}$ was generated by $\mathcal{B}$.

We now show how the challenger $\mathcal{B}$ translates $\mathcal{A}$'s forgery as a forgery with respect to the FKPR scheme SIG. If $\mathcal{B}$ has flipped $c = 0$ which means that $\mathrm{PK}^* := \mathrm{PK}'$, then the first three cases correspond to the forgeries where $\mathcal{A}$ has forged a signature under the secret key SK', and hence $\mathcal{B}$ can translate them to a forged signature corresponding to the scheme SIG which can be analogous to that in the proof of Theorem 1. Otherwise, if $\mathcal{A}$ outputs a forgery that belongs to the last two cases, $\mathcal{B}$ will abort.

If $c = 0$ which means that $\mathcal{B}$ sets PK' as the i^*th fresh proxy public key: in this case, if $\mathcal{A}$ outputs a forgery that belongs to the first three cases, then $\mathcal{B}$ will abort. Otherwise, the last two cases indicate that $\mathcal{A}$ has forged a signature under one of the keys generated by $\mathcal{B}$ in a delegation, but for which $\mathcal{A}$ has not received the corresponding secret key. In those two cases, $P\Sigma^*$ will be a valid signature under a key PK generated by $\mathcal{B}$ in some delegation query; that is, PK will be the last key in the list $\mathcal{FK}$ for a proxy key $(\mathcal{FK}, \mathsf{cert}, \mathrm{SK}_{\mathrm{prx}})$ from some proxy key list PskList$(*, *)$. Therefore, with probability $1/q_d$, $\mathcal{B}$ can choose the right i^* such that $\mathrm{PK} = \mathrm{PK}'$. In this case, $\mathcal{B}$ outputs $P\Sigma^*$ as a valid forgery of the key PK' for the underlying signature scheme SIG.

From the above analysis, we can see that the challenger $\mathcal{B}$'s forgery with a nonnegligible probability is contradictory to the security of the FKPR scheme SIG (cf. Theorem 1 of [37]) and thus proves the security of the LRPS scheme SIG^{**}.

Disclosure

An abstract of this paper has been presented in the proceedings of the 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS), IEEE, pp, 495–502, 2013 [42].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant no. 60970139), the Strategic Priority Program of Chinese Academy of Sciences (Grant no. XDA06010702), and the IIEs Cryptography Research Project. The authors would like to thank anonymous reviewers for their helpful comments and suggestions.

References

[1] W. Farmer, J. Gutmann, and V. Swarup, "Security for mobile agents: authentication and state appraisal," in *Computer Security—ESORICS 96: 4th European Symposium on Research in Computer Security Rome, Italy, September 25-27, 1996 Proceedings*, vol. 1146 of *Lecture Notes in Computer Science*, pp. 118–130, Springer, Berlin, Germany, 1996.

[2] P. Kotzanikolaous, G. Katsirelos, and V. Chrissikopoulos, "Mobile agents for secure electronic transactions," in *Recent Advances in Signal Processing and Communications*, pp. 363–368, World Scientific and Engineering Society Press, 1999.

[3] B. Lee, H. Kim, and K. Kim, "Secure mobile agent using strong non-designated proxy signature," in *Information Security and Privacy: Proceedings of the 6th Australasian Conference (ACISP '01), Sydney, Australia, July 11–13, 2001*, vol. 2119 of *Lecture Notes in Computer Science*, pp. 474–486, Springer, Berlin, Germany, 2001.

[4] B. Lee, H. Kim, and K. Kim, "Strong proxy signature and its applications," in *Proceedings of the Symposium on Cryptography and Information Security (SCIS '01)*, pp. 603–608, 2001.

[5] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: delegation of the power to sign messages," *IEICE Transactions on Fundamentals of Electronics*, vol. 79, pp. 1338–1353, 1996.

[6] G. Allée, S. Pierre, R. H. Glitho, and A. El Rhazi, "An improved itinerary recording protocol for securing distributed architectures based on mobile agents," *Mobile Information Systems*, vol. 1, no. 2, pp. 129–147, 2005.

[7] R. Aversa, B. Di Martino, N. Mazzocca, and S. Venticinque, "A skeleton based programming paradigm for mobile multi-agents on distributed systems and its realization within the MAGDA mobile agents platform," *Mobile Information Systems*, vol. 4, no. 2, pp. 131–146, 2008.

[8] K. Goto, Y. Sasaki, T. Hara, and S. Nishio, "Data gathering using mobile agents for reducing traffic in dense mobile wireless sensor networks," *Mobile Information Systems*, vol. 9, no. 4, pp. 295–314, 2013.

[9] Y. Wang, D. S. Wong, and H. Wang, "Employ a mobile agent for making a payment," in *Mobile Information Systems*, vol. 4, pp. 51–68, IOS Press, 2008.

[10] S. Parvin, F. K. Hussain, and S. Ali, "A methodology to counter DoS attacks in mobile IP communication," *Mobile Information Systems*, vol. 8, no. 2, pp. 127–152, 2012.

[11] H. U. Park and I. Y. Lee, "A digital nominative proxy signature scheme for mobile communication," in *Information and Communications Security: Third International Conference, ICICS 2001 Xian, China, November 13–16, 2001 Proceedings*, vol. 2229 of *Lecture Notes in Computer Science*, pp. 451–455, Springer, Berlin, Germany, 2001.

[12] S. Kim, S. Park, and D. Won, "Proxy signatures, revisited," in *Proceedings of the 1st International Conference on Information and Communication Security (ICICS '97)*, vol. 1334 of *Lecture Notes in Computer Science*, pp. 223–232, Springer, 1997.

[13] A. Boldyreva, A. Palacio, and B. Warinschi, "Secure proxy signature schemes for delegation of signing rights," *Journal of Cryptology*, vol. 25, no. 1, pp. 57–115, 2012.

[14] T. Malkin, S. Obana, and M. Yung, "The hierarchy of key evolving signatures and a characterization of proxy signatures," in *Advances in Cryptology—EUROCRYPT 2004*, vol. 3027 of *Lecture Notes in Computer Science*, pp. 306–322, Springer, Berlin, Germany, 2004.

[15] J. C. N. Schuldt, K. Matsuura, and K. G. Paterson, "Proxy signature secure against key exposure," in *Public Key Cryptography—PKC 2008: 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings*, vol. 4939 of *Lecture Notes in Computer Science*, pp. 141–161, Springer, Berlin, Germany, 2008.

[16] H. Wang and J. Pieprzyk, "Efficient one-time proxy signatures," in *Advances in Cryptology—ASIACRYPT 2003*, vol. 2894 of *Lecture Notes in Computer Science*, pp. 507–522, Springer, Berlin, Germany, 2003.

[17] F. Zhang, R. Safavi-Naini, and C. Y. Lin, "New proxy signature, proxy blind signature and proxy ring signature schemes from bilinear pairings," Tech. Rep. 2003/104, Cryptology ePrint Archive, 2003, http://eprint.iacr.org/.

[18] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[19] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegating signing operation," in *Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS '96)*, pp. 48–56, ACM, March 1996.

[20] J. Y. Lee, J. H. Cheon, and S. Kim, "An analysis of proxy signatures: is a secure channel necessary?" in *Proceedings of the Cryptographers' Track at the RSA Conference, San Francisco, Calif, USA, April 2003*, Lecture Notes in Computer Science, pp. 68–79, Springer, 2003.

[21] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Strong key-insulated signature schemes," in *Public Key Cryptography—PKC 2003*, vol. 2567 of *Lecture Notes in Computer Science*, pp. 130–144, Springer, Berlin, Germany, 2002.

[22] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Computer Networks*, vol. 48, no. 5, pp. 701–716, 2005.

[23] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO'99*, vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397, Springer, Berlin, Germany, 1999.

[24] E. Biham, Y. Carmeli, and A. Shamir, "Bug attacks," in *Advances in Cryptology—CRYPTO 2008*, vol. 5157 of *Lecture Notes in Computer Science*, pp. 221–240, Springer, Berlin, Germany, 2008.

[25] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Advances in Cryptology—EUROCRYPT'97*, vol. 1233 of *Lecture Notes in Computer Science*, pp. 37–51, Springer, Berlin, Germany, 1997.

[26] S. Micali and L. Reyzin, "Physically observable cryptography," in *Theory of Cryptography: Proceedings of the 1st Theory of Cryptography Conference (TCC '04), Cambridge, MA, USA, February 19—21, 2004*, vol. 2951 of *Lecture Notes in Computer Science*, pp. 278–296, Springer, Berlin, Germany, 2004.

[27] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan, "Overcoming the hole in the bucket: public-key cryptography resilient to continual memory leakage," in *Proceedings of the IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS '10)*, pp. 501–510, October 2010.

[28] Y. Dodis, K. Haralambiev, A. Lopez-Alt, and D. Wichs, "Cryptography against continuous memory attacks," in *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, pp. 511–520, 2010.

[29] K. Pietrzak, "A leakage-resilient mode of operation," in *Advances in Cryptology—EUROCRYPT '09*, vol. 5479 of *Lecture Notes in Computer Science*, pp. 462–482, Springer, Berlin, Germany, 2009.

[30] S. Garg, A. Jain, and A. Sahai, "Leakage-resilient zero knowledge," in *Advances in Cryptology—CRYPTO 2011*, vol. 6841 of *Lecture Notes in Computer Science*, pp. 297–315, Springer, Berlin, Germany, 2011.

[31] E. Kiltz and K. Pietrzak, "Leakage resilient ElGamal encryption," in *Advances in Cryptology—ASIACRYPT '10*, vol. 6477 of *Lecture Notes in Computer Science*, pp. 595–612, Springer, Berlin, Germany, 2010.

[32] M. Naor and G. Segev, "Public-key cryptosystems resilient to key leakage," in *Advances in Cryptology—CRYPTO 2009*, vol. 5677 of *Lecture Notes in Computer Science*, pp. 18–35, Springer, Berlin, Germany, 2009.

[33] S. S. M. Chow, Y. Dodis, Y. Rouselakis, and B. Waters, "Practical leakage-resilient identity-based encryption from simple assumptions," in *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*, pp. 152–161, ACM, October 2010.

[34] T. H. Yuen, S. S. M. Chow, Y. Zhang, and S. M. Yiu, "Identity-based encryption resilient to continual auxiliary leakage," in

Advances in Cryptology—EUROCRYPT 2012, vol. 7237 of *Lecture Notes in Computer Science*, pp. 117–134, Springer, Berlin, Germany, 2012.

[35] J. Alwen, Y. Dodis, and D. Wichs, "Leakage-resilient public-key cryptography in the bounded-retrieval model," in *Advances in Cryptology—CRYPTO 2009*, vol. 5677 of *Lecture Notes in Computer Science*, pp. 36–54, Springer, 2009.

[36] E. Boyle, G. Segev, and D. Wichs, "Fully leakage-resilient signatures," in *Advances in Cryptology—EUROCRYPT 2011*, vol. 6632 of *Lecture Notes in Computer Science*, pp. 89–108, Springer, Berlin, Germany, 2011.

[37] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum, "Leakage-resilient signatures," in *Theory of Cryptography: 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, vol. 5978 of *Lecture Notes in Computer Science*, pp. 343–360, Springer, Berlin, Germany, 2010.

[38] J. Katz and V. Vaikuntanathan, "Signature schemes with bounded leakage resilience," in *Advances in Cryptology—ASIACRYPT 2009*, vol. 5912 of *Lecture Notes in Computer Science*, pp. 703–720, Springer, Berlin, Germany, 2009.

[39] T. Malkin, I. Teranishi, Y. Vahlis, and M. Yung, "Signatures resilient to continual leakage on memory and computation," in *Proceedings of the 8th Theory of Cryptography Conference (TCC '11)*, vol. 6597 of *Lecture Notes in Computer Science*, pp. 89–106, Springer, Providence, RI, USA, 2011.

[40] F. Tang, H. Li, Q. Niu, and B. Liang, "Efficient leakage-resilient signature schemes in the generic bilinear group model," Cryptology ePrint Archive 2013/785, 2013, http://eprint.iacr.org/.

[41] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology—EUROCRYPT 2003*, vol. 2656 of *Lecture Notes in Computer Science*, pp. 416–432, Springer, Berlin, Germany, 2003.

[42] F. Tang, H. Li, Q. Niu, and B. Liang, "Leakage-resilient proxy signatures," in *Proceedings of the 5th IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS '13)*, pp. 495–502, Xi'an, China, September 2013.