2013 IEEE 27th International Conference on Advanced Information Networking and Applications

A Generic Construction of Proxy Signatures from Certificateless Signatures

Lei Zhang[1], Qianhong Wu[2,3], Bo Qin[2,4], Josep Domingo-Ferrer[2], Peng Zeng[1], Jianwei Liu[5], Ruiying Du[3]
[1]Shanghai Key Laboratory of Trustworthy Computing
Software Engineering Institute, East China Normal University, Shanghai, China
[2]UNESCO Chair in Data Privacy, Department of Computer Engineering and Mathematics
Universitat Rovira i Virgili, Tarragona, Catalonia
[3]Key Lab. of Aerospace Information Security and Trusted Computing Ministry of Education
Wuhan University, School of Computer, China
[4]Department of Maths, School of Science, Xi'an University of Technology, China
[5]School of Electronic and Information Engineering, Beijing University of Aeronautics & Astronautics, China
leizhang@sei.ecnu.edu.cn, {qianhong.wu,bo.qin,josep.domingo}@urv.cat, pzeng@sei.ecnu.edu.cn,
liujianwei@buaa.edu.cn, duraying@126.com

***Abstract*—The primitive of proxy signatures allows the original signer to delegate proxy signers to sign on messages on behalf of the original signer. It has found numerous applications in distributed computing scenarios where delegation of signing rights is common. Certificateless public key cryptography eliminates the complicated certificates in traditional public key cryptosystems without suffering from the key escrow problem in identity-based public key cryptography. In this paper, we reveal the relationship between the two important primitives of proxy signatures and certificateless signatures and present a generic conversion from the latter to the former. Following the generic transformation, we propose an efficient proxy signature scheme with a recent certificateless signature scheme.**

Index Terms — certificateless public key cryptography, proxy signature, certificateless signature, provable security.

I. Introduction

The concept of proxy signatures was first introduced by Mambo *et al.* [11]. It allows a user, called original signer, to delegate another user, called proxy signer, to sign messages on behalf of the original signer, in case of say, temporal absence, lack of time or computational power, *etc.*. The proxy signature schemes [15], [16], [19] have found lots of practical applications in distributed systems, grid computing, mobile agent applications, *etc.*, where delegation of signing rights is quite common. Based on the delegation types, proxy signatures can be classified into three types: full delegation, partial delegation and delegation by warrant, in which the last one attracts most interest. One of the main reasons is that it is flexible for the original signer to include any type of security policy (that specifies what kinds of messages are delegated, and may contain other information, such as the identities of the original signer and the proxy signer, the delegation period, *etc.*) in the warrant to describe the restrictions for the proxy signer under which the delegation is valid.

Following Mambo *et al.* [11], many proxy signature schemes have been proposed. However, most of them does not have provable-security guarantees[4], [5]. The security model of proxy signatures was studied in [3] for the first time, and later in [10], [13]. In [3], [13], the authors also discussed the relation of proxy signatures and other signatures. They showed that proxy signatures could be constructed from standard signatures and aggregate signatures. Malkin *et al.* [7] showed that proxy signatures are equivalent to key-insulated signatures [6].

To let a digital signature scheme work properly, the signature scheme in traditional public key cryptosystems requires that a user must be bounded with its public key. This binding is provided by a public key infrastructure in which a trusted certificate authority issues public key certificates that securely bind the users to their public keys. In practice, the management of public key certificates requires is a heavy burden in terms of computation, distribution and storage. Identity-based public key cryptography (ID-PKC) [12] was suggested to reduce such cost. In this approach, a user's identity such as the telephone number or email address plays the role of a public key in the traditional public key cryptosystems; while the private key associating with the identity of the user is generate by a trusted authority referred to as private key generator (PKG). The recognizable identity feature of ID-PKC eliminates the heavy certificate management burden in the traditional public key cryptosystems. The drawback of ID-PKC is that it suffers from the key escrow problem. That is, the PKG knows the private keys of all the users in the system and it can forge any signature of any user in the system without being detected. Thus, ID-PKC can only be applicable in the case that the PKG can be fully trusted by the users.

Certificateless public key cryptography (CL-PKC) [1] was introduced to solve the key escrow problem in ID-PKC, while keeping the implicit certification of a user's public key. CL-PKC also employs an authority called key generate center (KGC) to help a user to generate his/her private key. However, the KGC only generates a partial private key for a user. The full private key of the user is combined by his/her partial private key generated by the KGC and a secret value chosen by himself/herself. Since the KGC does not access to the full

1550-445X/13 $26.00 © 2013 IEEE
DOI 10.1109/AINA.2013.54

private key of any user, CL-PKC overcomes the key escrow problem which is inherent in ID-PKC systems.

Certificateless signatures have recently been intensively investigated in recent years. The first certificateless signature scheme was present in [1]. Since then, several certificateless signature schemes [8], [9], [14], [17], [18] have been proposed. In [9], Huang *et al.* revisited the security models of certificateless signature schemes. They classified the adversary in CL-PKC into three types, namely the normal, strong and super Type I/II adversaries (See Section III-B). Among them, the last type of adversaries is the most powerful one. Thus, it is enjoyable if certificateless signature schemes [8], [9], [17], [18] can be proven secure against super Type I/II adversaries.

Our Contribution. In this paper, we investigate the relationship between the two cryptographic primitives of certificateless signatures and proxy signatures. Specifically, we show that any certificateless signature scheme secure against super Type I and II adversaries can be converted into a secure warrant-based proxy signature scheme which is the most flexible type in practice. The generic conversion enjoys a tight security reduction. Finally, a concrete proxy signature scheme based on a recent certificateless signature scheme [18] from bilinear map is instantiated. Bilinear map operation is much more expensive than other cryptographic operations in bilinear map based cryptosystems. The resulting proxy signature scheme is bilinear map operation free at the signature generation stage and only requires to operate one bilinear map operation at the signature verification stage. Therefore, the proxy signature scheme achieves almost the best efficiency in bilinear map based cryptosystem.

Paper Organization. In Section II, we give the notion of proxy signature schemes and the adversarial model of proxy signature schemes. Section III defines the notion of certificateless signature schemes and the corresponding adversarial model. We present our generic construction of proxy signature schemes from certificateless signature schemes in Section IV. A concrete instantiation is given in Section V. Finally, Section VI concludes the paper.

II. Proxy Signature Schemes

In this section, we first review the notion of proxy signature schemes and then give the adversarial model for proxy signature schemes.

A. Syntax of Proxy Signature Schemes

A proxy signature scheme is a digital signature scheme comprised of the following algorithms:

- GlobeSetup(k): This algorithm accepts a security parameter k and returns a list of system common parameters *params* (such as the descriptions of the groups, hush functions, etc.).
- OKeyGen(*params*): This algorithm takes as input *params*, and generates the private/public key s_o/P_o of an original signer.
- PKeyGen($params, P_o$[1]): This algorithm takes as input *params*, an original signer's public key P_o and generates the private/public key s_p/P_p of a proxy signer.
- Delegate($params, P_o, s_o, m_w$): This algorithm takes as input *params*, an original signer's private/public key s_o/P_o, a warrant m_w[2] and outputs a delegation $\varpi = (m_w, \sigma_w)$.
- DVerify($params, P_o, \varpi$): This algorithm takes as input *params*, an original signer's public key P_o, a delegation ϖ and verifies whether ϖ is a valid delegation. It outputs *true* if the delegation is valid or *false* otherwise.
- PKGen($params, P_o, \varpi, s_p$): This is the proxy signing key generation algorithm which takes as input *params*, an original signer's public key P_o, a delegation ϖ and the private key of a proxy signer s_p. It outputs a proxy signing key K_p.
- PSign($params, P_o, m_w, P_p, K_p, m$): This is the proxy signing algorithm which takes as input *params*, P_o, m_w, P_p, a proxy signing key K_p and a message m. It outputs a proxy signature σ.
- PVerify($params, P_o, P_p, m_w, m, \sigma$): This is a proxy signature verification algorithm which takes as input *params*, the original signer's public key P_o, a warrant m_w, a message m, the proxy signer's public key P_p and a proxy signature σ. It outputs *true* if the proxy signature is valid or *false* otherwise.

B. Adversarial Model for Proxy Signature Schemes

To discuss the security of a proxy signature scheme, as defined in [10], we can classify the adversaries into following three types:

- Type $\mathbb{A}$: This adversary has the public keys of the original signer and the proxy signer and also has the private key of the proxy signer.
- Type $\mathbb{B}$: This adversary has the public keys of the original signer and the proxy signer and also has the private key of the original signer.
- Type $\mathbb{C}$: This adversary only has the public keys of the original signer and the proxy signer.

It is easy to see that if a proxy signature scheme is secure against a Type $\mathbb{A}$ or $\mathbb{B}$ adversary, then the scheme is also secure against a Type $\mathbb{C}$ adversary. Therefore, for a secure proxy signature scheme, we only need to prove that the scheme is secure against Type $\mathbb{A}$ and $\mathbb{B}$ adversaries.

To define the security of a proxy signature scheme, we demonstrate two games played between a challenger $\mathcal{C}$ and a Type $\mathbb{A}$ adversary $\mathcal{A}_I$ or a Type $\mathbb{B}$ adversary $\mathcal{A}_{II}$.

Game 1 for Type $\mathbb{A}$ adversary.

$\underline{Setup}$: $\mathcal{C}$ first runs GlobeSetup(k) to generate the system common parameters *params*; then runs OKeyGen(*params*) to generate the private-public key pair (s_o, P_o) for the original

[1] P_o is an optical input for this algorithm.

[2] It may contain the identity of the designated proxy signer and, possibly, restrictions on the message that the proxy signer is allowed to sign.

signer; and then runs PKeyGen($params, P_o$) to generate the private and public key pair (s_p, P_p) for the proxy signer. Finally, $\mathcal{C}$ sends *params*, P_o and (s_p, P_p) to the adversary $\mathcal{A}_I$.

Attack : $\mathcal{A}_I$ has the ability to access the following oracles (as well as the random orales [2] if there exist) which are controlled by $\mathcal{C}$.

- Delegate-Oracle($params, P_o, m_w$): On input *params*, the original signer's public key P_o and a warrant m_w, this oracle outputs a delegation ϖ.
- PSign-Oracle($params, P_o, m_w, P_p, m$): This oracle accepts *params*, the original signer's public key P_o, a warrant m_w, the proxy signer's public key P_p, a message m, and outputs a proxy signature σ.

Forgery : Finally, $\mathcal{A}_I$ outputs a tuple (m_w^*, m^*, σ^*). We say $\mathcal{A}_I$ wins Game 1, if the following cases are satisfied:

1) σ^* is a valid proxy signature on m under P_o, P_p and m_w.
2) $(params, P_o, m_w^*)$ has never been submitted to the Delegate-Oracle.
3) $\mathcal{A}_I$ has never submitted $(params, P_o, m_w^*, P_p, m^*)$ to the PSign-Oracle.

Definition 1: A proxy signature scheme is existentially unforgeable against $\mathcal{A}_I$ under adaptively chosen-message attacks if and only if the probability of success of any polynomially bounded $\mathcal{A}_I$ in the above game is negligible.

Game 2 for Type $\mathbb{B}$ adversary.

Setup : $\mathcal{C}$ first runs GlobeSetup(k) to obtain the system common parameters *params* then runs OKeyGen(*params*) to generate the private-public key pair (s_o, P_o) for the original signer and then runs PKeyGen($params, P_o$) to generate the private-public key pair (s_p, P_p) for the proxy signer. Eventually, $\mathcal{C}$ sends *params* (s_o, P_o) and P_p to $\mathcal{A}_{II}$.

Attack : $\mathcal{A}_{II}$ has the ability to access the following PSign-Oracle (as well as the random orales if there exists) which is controlled by $\mathcal{C}$. Note that the Delegate-Oracle is not required, since $\mathcal{A}_{II}$ knows the original signer's private key s_o.

- PSign-Oracle($params, P_o, m_w, P_p, m$): This oracle accepts *params*, the original signer's public key P_o, a warrant m_w, the proxy signer's public key P_p, a message m and outputs a proxy signature σ.

Forgery : Finally, $\mathcal{A}_{II}$ outputs a tuple (m_w^*, m^*, σ^*). We say $\mathcal{A}_{II}$ wins Game 2, if the following cases are satisfied:

1) σ^* is a valid proxy signature under P_o, P_p and m_w.
2) $\mathcal{A}_{II}$ has never submitted $(params, P_o, m_w^*, P_p, m^*)$ to the PSign-Oracle.

Definition 2: A proxy signature scheme is existentially unforgeable against $\mathcal{A}_{II}$ under adaptively chosen-message attacks if and only if the probability of success of any polynomially bounded $\mathcal{A}_{II}$ in the above game is negligible.

III. Certificateless Signature Schemes

In this section, we first review the notion of certificateless signature schemes, then we introduce the security model for certificateless signature schemes.

A. Notion of Certificateless Signature Schemes

A certificateless signature scheme consists of eight algorithms. The description of each algorithm is as follows:

- Setup(k): It consists of following two sub-algorithms *ParamGen* and *MKGen*:
  - *ParamGen*: On input a security parameter k, this algorithm first generates a list of partial system parameters $params$ (such as the descriptions of the groups, hush functions, etc.).
  - *MKGen*: It accepts $params$ to generate the KGC's master key *master-key* and master public key *master-public-key*.

  Let $params' = (params,$ *master-public-key*$)$ be the full system parameters. $params'$ is published in the system while *master-key* is kept secretly.
- Partial-Private-Key-Extract($params', master\text{-}key, ID$): This algorithm takes as input $params'$, *master-key*, a user's identity ID, and produces the user's partial private key D_{ID}.
- Partial-Private-Key-Verify($params', ID, D_{ID}$)[3]: This algorithm accepts $params'$, a user's identity ID and partial private key D_{ID} to check the validity of the partial private key.
- Set-Secret-Value($params'$): This algorithm accepts $params'$ to produce a secret value x for a user.
- Set-Public-Key($params', x$): This algorithm accepts $params'$, a user's secret value x to produce a public key P_{ID} for the user.
- Set-Private-Key($params', ID, D_{ID}, x$)[4]: This algorithm accepts $params'$, a user's identity ID, partial private key D_{ID} and secret value x to produce a private signing key S_{ID} for the user.
- Sign($params', m, ID, P_{ID}, S_{ID}$): This algorithm takes as input $params'$, a message m, a signer's identity ID, public key P_{ID} and private key S_{ID}, and outputs a signature σ.
- Verify($params', m, ID, P_{ID}, \sigma$): This algorithm takes as input $params'$, m, ID, P_{ID}, a signature σ and verifies whether the signature is valid or not. It outputs *true* if the signature is valid or *false* otherwise.

B. Adversarial Model for Certificateless Signature Schemes

There are two types of adversaries [1] with different capabilities that are generally considered in CL-PKC. They are known as *Type I Adversaries* and *Type II Adversaries*. A *Type I Adversary* $\mathcal{B}_I$ does not have access to the *master-key*, but he has the ability to replace the public key of any user with a value of his choice. While a *Type II Adversary* $\mathcal{B}_{II}$ has access to the *master-key* but cannot perform public key replacement. Obviously, a secure certificateless signature

[3]This algorithm is usually omitted in a certificateless signature scheme. Because there is an assumption that the KGC always generates the private keys honestly.

[4]This algorithm is sometimes omitted in some certificateless signature schemes. This is because, knowing the secret value and partial private key, a user can compute its private key easily.

scheme must avoid both types of the adversaries to forge a valid certificateless signature. In [9], the Type I/II adversaries are further classified into three types, i.e., normal, strong and super Type I/II adversaries, in which a super Type I/II adversary is the strongest Type I/II adversary in CL-PKC. By the setting in [9], a super Type I/II adversary can obtain some message-signature pairs which are valid under the public key chosen by the adversary himself even he does know the secret key corresponding to the public key. In this paper, we treat $\mathcal{B}_I$ and $\mathcal{B}_{II}$ as super adversaries.

To define the security of a certificateless signature scheme, we demonstrate two games played between a challenger $\mathcal{C}$ and an adversary $\mathcal{B}_I$ or $\mathcal{B}_{II}$.

Game 3 for Type I Adversary

$\underline{Setup}$: $\mathcal{C}$ runs Setup(k) to obtain the partial system parameter list *params*, the *master-key* and the *master-public-key*. Let $params' = (params,$ *master-public-key*$)$. $\mathcal{C}$ sends $params'$ to $\mathcal{B}_I$.

$\underline{Attack}$: $\mathcal{B}_I$ has the ability to access the following oracles (as well as the random orales if there exists) which are controlled by $\mathcal{C}$.

- Partial-Private-Key-Oracle$(params', ID)$: On input $params'$ and an identity ID, it outputs a partial private key D_{ID} of the user whose identity is ID.
- Public-Key-Oracle$(params', ID)$: On input $params'$ and an identity ID, it outputs the public key P_{ID} associated with the identity.
- Secret-Value-Oracle$(params', ID)$: On input $params'$ and an identity ID, it outputs the secret value x associated with the user's identity. It outputs $\perp$, if the user's public key has been replaced.
- Public-Key-Replacement-Oracle$(params', ID, P'_{ID})$: On input $params'$, an identity ID and a new public key P'_{ID}, it replaces the associated user's public key with the new one.
- Sign-Oracle$(params', m, ID, P_{ID})$: On input $params'$, a message m, an identity ID and a public key P_{ID}, it outputs a signature σ.

$\underline{Forgery}$: Finally, $\mathcal{B}_I$ outputs a tuple $(ID^*, m^*, \sigma^*, P_{ID^*})$. We say $\mathcal{A}$ wins the game, if all of the following cases are satisfied:

1) σ^* is a valid signature produced by $\mathcal{B}_I$.
2) $\mathcal{B}_I$ has not submitted $(params', ID^*)$ to the Partial-Private-Key-Oracle.
3) $(params', m^*, ID^*, P_{ID^*})$ has never been submitted to the Sign-Oracle.

Definition 3: A certificateless signature scheme is existentially unforgeable against $\mathcal{B}_I$ under adaptively chosen-message attacks if and only if the probability of success of any polynomially bounded $\mathcal{B}_I$ in the above game is negligible.

Game 4 for Type II Adversary

$\underline{Setup}$: $\mathcal{C}$ runs Setup(k) to obtain the partial system parameter list *params*, *master-key* and *master-public-key*. Let $params' =$ $(params,$ *master-public-key*$)$. Finally, $\mathcal{C}$ sends $params'$ and *master-key* to $\mathcal{B}_{II}$.

$\underline{Attack}$: $\mathcal{B}_{II}$ has the ability to access the following oracles (as well as the random orales if there exists) which are controlled by $\mathcal{C}$.

- Public-Key-Oracle$(params', ID)$: On input $params'$ and an identity ID, it outputs the public key P_{ID} associated with ID.
- Secret-Value-Oracle$(params', ID)$: On input $params'$ and an identity ID, it outputs the secret value x associated with ID. It outputs $\perp$, if the user's public key has been replaced.
- Public-Key-Replacement-Oracle$(params', ID, P'_{ID})$: On input $params'$, an identity ID and a new public key P'_{ID}, it replaces the associated user's public key with the new one.
- Sign-Oracle$(params', ID, P_{ID}, m)$: On input $params'$, an identity ID, a public key P_{ID} and a message m, it outputs a signature σ.

$\underline{Forgery}$: Finally, $\mathcal{B}_{II}$ outputs a tuple $(ID^*, m^*, \sigma^*, P_{ID^*})$. We say $\mathcal{B}_{II}$ wins the game, if following cases are satisfied:

1) σ^* is a valid signature produced by $\mathcal{B}_{II}$.
2) $\mathcal{B}_{II}$ has not submitted $(params', ID)$ to the Secret-Value-Oracle.
3) $(params', ID^*, P'_{ID^*})$ has never queried to the Public-Key-Replacement-Oracle.
4) $(params', m^*, ID^*, P_{ID^*})$ has never been submitted to the Sign-Oracle.

Definition 4: A certificateless signature scheme is existentially unforgeable against $\mathcal{B}_{II}$ under adaptively chosen-message attacks if and only if the probability of success of any polynomially bounded $\mathcal{B}_{II}$ in the above game is negligible.

IV. Proxy Signatures from Certificateless Signatures

In this section, we present a generic construction of proxy signatures from certificateless signatures. The construction is as follows.

A. Proxy Signatures from Certificateless Signatures

Let Ω be a certificateless signature scheme which consist of Setup(k), Partial-Private-Key-Extract($params'$, $master\text{-}key, ID$), Partial-Private-Key-Verify($params'$, ID, D_{ID}), Set-Secret-Value($params'$), Set-Private-Key(*pa-rams'*, ID, D_{ID}, x), Set-Public-Key($params', x$), Sign($params'$, ID, P_{ID}, S_{ID}, m) and Verify(*params'*, *ID*, P_{ID}, m, σ). We now show how to convert Ω to be a proxy signature scheme. The conversion comes as follows:

- GlobeSetup(k): On input a security parameter k, this algorithm runs Setup(k) of Ω, except that it doesn't run '*MKGen*', to generate the system common parameters $params$.
- OKeyGen($params$): This algorithm first runs '*MKGen*' which accepts $params$ to generate a master key *master-key* and master public key *master-public-key*; then sets

the original signer's private key $s_o =$ *master-key*, public key $P_o =$ *master-public-key*.

Let $params' = (params, P_o)$. Note P_o is an optical input for the algorithm PKeyGen($params'$).

- PKeyGen($params'$): This algorithm first runs Set-Secret-Value($params'$) of Ω to generate a secret value x then runs Set-Public-Key($params', x$) of Ω to generate a public key P_x. Finally, this algorithm sets the proxy signer's private key $s_p = x$, public key $P_p = P_x$.
- Delegate($params', s_o, m_w$): On input a warrant m_w, the original signer whose private key is s_o, runs Partial-Private-Key-Extract($params', s_o, m_w$) of Ω to generate a delegation $\varpi = (m_w, \sigma_w)$.
- DVerify($params', \varpi$): To verify the validity of a delegation $\varpi = (m_w, \sigma_w)$ from the original signer, the proxy signer runs Partial-Private-Key-Verify($params', m_w, \sigma_w$) of Ω.
- PKgen($params', \varpi, x_p$): The proxy signer whose private is s_p, accepts $\varpi = (m_w, \sigma_w)$, runs Set-Private-Key($params', m_w, \sigma_w, x_p$) of Ω to obtain a private key S_w, and sets the proxy signing key $K_p = S_w$.
- PSign($params', m_w, P_p, K_p, m$): Let K_p be the proxy signer's proxy signing key. The proxy signer runs Sign($params', m_w, P_p, K_p, m$) of Ω to generate a proxy signature σ.
- PVerify($params', m_w, m, \sigma, P_p$): To verify the validity of a proxy signature (m_w, m, σ), a verifier first checks whether the proxy signer and the message conform to m_w, if so runs Verify($params', m_w, P_p, m, \sigma$) of Ω; otherwise, this algorithm outputs $false$.

B. Security Proofs

In this section, we show the security of above proxy signature scheme.

Theorem 1: The proposed proxy signature scheme is secure against type $\mathbb{A}$ adversary $\mathcal{A}_I$ (as defined in Game 1 in Section II-B) if the underlining certificateless signature scheme is existentially unforgeable against Type I adversary $\mathcal{B}_I$ under adaptively chosen-message attacks.

Proof: Let $\mathcal{C}$ be the challenger of the certificateless signature scheme, $\mathcal{B}_I$ be a type I adversary who also acts as the challenger of the proxy signature scheme, $\mathcal{A}_I$ be a type $\mathbb{A}$ adversary who can break our proxy signature scheme in time τ with advantage ϵ. $\mathcal{C}$ controls Partial-Private-Key-Oracle, Public-Key-Oracle, Secret-Value-Oracle, Public-Key-Replacement-Oracle and Sign-Oracle while $\mathcal{B}_I$ controls Delegate-Oracle and PSign-Oracle. In the following, we show that if there's a type $\mathbb{A}$ adversary $\mathcal{A}_I$ can break the proxy signature scheme in time τ with advantage ϵ, then $\mathcal{B}_I$ can use $\mathcal{A}_I$ to break the underlining certificateless signature scheme in time $\mathcal{O}(\tau)$ with advantage ϵ.

Setup : $\mathcal{C}$ selects the system parameters $params$ and *master-public-key*, sends $params' =$ ($params$, *master-public-key*) to $\mathcal{B}_I$. $\mathcal{B}_I$, after receiving $params'$ from $\mathcal{C}$, sets the original signer's public key as $P_o =$ *master-public-key*, chooses a secret value x_p as the proxy signer's private key and generates the proxy signer's public key P_p, sends $params, P_o, x_p, P_p$ to $\mathcal{A}_I$.

Attack : $\mathcal{A}_I$ can ask $\mathcal{B}_I$ following oracle queries.

Delegate-Oracle($params, m_w$) queries: On input $params$ and a warrant m_w, $\mathcal{B}_I$ submits ($params', m_w$) to the Partial-Private-Key-Oracle controlled by $\mathcal{C}$ for a partial private key D_{m_w}, where $params' =$ ($params$, *master-public-key*). Finally, $(m_w, \sigma_w = D_{m_w})$ is returned to $\mathcal{A}_I$.

PSign-Oracle($params, m_w, m$) queries: On input $(params, m_w, m)$, $\mathcal{B}_I$ first submits $(params', m_w, P_p, m)$ to the Sign-Oracle to generate a certificateless signature σ, then returns (m_w, m, σ) to $\mathcal{A}_I$.

Forgery : At the end of $\mathcal{A}_I$'s attack, he outputs a tuple $\{m_w^*, m^*, \sigma^*\}$. $\mathcal{B}_I$ sets $ID^* = m_w^*$, outputs $(ID^*, m^*, \sigma^*, P_p)$ as a certificateless signature forgery. It is easy to see that σ^* is a valid certificateless signature under the identity ID^* and public key P_p, since PVerify($params', m_w, m, \sigma, P_p$)= Verify($params', m_w, P_p, m, \sigma$)=$ture$.

In above simulation, $\mathcal{A}_I$'s view is identical to his view in the real attack. Therefore, the success probability of $\mathcal{A}_I$ is also ϵ and $\mathcal{A}_I$'s running time is $\mathcal{O}(\tau)$. ■

Theorem 2: The proposed proxy signature scheme is secure against type $\mathbb{B}$ adversary $\mathcal{A}_{II}$ (as defined in Game 2 in Section II-B) if the underlining certificateless signature scheme is existentially unforgeable against Type II adversary $\mathcal{B}_{II}$ under adaptively chosen-message attacks.

Proof: Let $\mathcal{C}$ be the challenger of the certificateless signature scheme, $\mathcal{B}_{II}$ be a type II adversary who also acts as the challenger of the proxy signature scheme, $\mathcal{A}_{II}$ be a type $\mathbb{B}$ adversary who can break our proxy signature scheme in time τ' with advantage ϵ'. $\mathcal{C}$ controls Partial-Private-Key-Oracle, Public-Key-Oracle, Secret-Value-Oracle, Public-Key-Replacement-Oracle and Sign-Oracle while $\mathcal{B}_{II}$ controls PSign-Oracle.

Setup : $\mathcal{C}$ generates the system parameters $params$, the *master-key* and *master-public-key*, sends *master-key*, $params' =$ ($params$, *master-public-key*) to $\mathcal{B}_{II}$. After receiving *master-key* and $params'$), $\mathcal{B}_{II}$ randomly choose an identity ID^* and submits ID^* to the Public-Key-Oracle which is controlled by $\mathcal{C}$ for a public key P_p^*. Finally, $\mathcal{B}_{II}$ sets (*master-key*, *master-public-key*) as the original signer's private and public keys, set P_p^* as the proxy signer's public key, sends ($params$,*master-key*, *master-public-key*) to $\mathcal{A}_{II}$.

Attack : As defined in Section III-B, $\mathcal{A}_{II}$ can ask $\mathcal{B}_{II}$ following PSign-Oracle queries.

PSign-Oracle($params, m_w, m$) queries: On input $(params, m_w, m)$, $\mathcal{B}_{II}$ submits $(params', m_w, P_p^*, m)$ to the Sign-Oracle to generate a certificateless signature σ, and returns (m_w, m, σ) to $\mathcal{A}_{II}$.

Forgery : Finally, $\mathcal{A}_{II}$ outputs a tuple $\{m^*, m_w^*, \sigma^*\}$. $\mathcal{B}_{II}$ sets $ID^* = m_w^*$, outputs $(ID^*, P_p, m^*, \sigma^*)$ as a certificateless

signature forgery. It is easy to see that σ^* is valid signature on m^* under ID^* and P_p.

In the above simulation, the $\mathcal{A}_{II}$'s view is identical to the view in the real attack. Therefore, the success probability of $\mathcal{A}_{II}$ is also ϵ and $\mathcal{A}_{II}$'s running time is $\mathcal{O}(\tau)$. ■

V. A Concrete Conversion

In this section, we present a concrete construction of proxy signature scheme from a certificateless signature scheme. The certificateless signature scheme we chosen is the certificateless signature scheme in [18] and can be also found in Appendix A. This scheme is one of the most efficient certificateless signature schemes secure against super types I and II adversaries.

A. Bilinear Maps

Nowadays, bilinear maps based cryptosystems have become a major research topic in cryptography. The scheme in [18] is based on bilinear maps. Thus, we first briefly review them. Let G_1 be an additive group of prime order p and G_2 be a multiplicative group of the same order. Let *P* denote a generator of G_1. A map $e : G_1 \times G_1 \longrightarrow G_2$ is called a bilinear map if it satisfies the following properties:

1) Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for $P, Q \in G_1, a, b \in Z_p^*$.
2) Non-degeneracy: There exists $P, Q \in G_1$ such that $e(P, Q) \neq 1$.
3) Computability: There exists an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in G_1$.

B. The Conversion

The concrete conversion comes as follows:

- GlobeSetup(k): This algorithm runs as follows:
  1) Select a cyclic additive group G_1 which is generated by P with prime order q, chooses a cyclic multiplicative group G_2 of the same order and a bilinear map $e : G_1 \times G_1 \longrightarrow G_2$.
  2) Set $g = e(P, P)$.
  3) Choose cryptographic hash functions $H_1 : \{0,1\}^* \longrightarrow Z_p^*$ and $H_2 : \{0,1\}^n \times G_2 \times G_2 \times G_2 \longrightarrow Z_p^*$, where *n* denote the bit-length of plaintexts.

  The system parameter list is $params = (G_1, G_2, e, P, g, H_1, H_2)$.
- OKeyGen($params$): This algorithm accepts $params$ and chooses a random $\lambda \in Z_q^*$ as the original signer's private key and sets $P_o = \lambda P$ as the original signer's public key.
- PKeyGen($params'$): This algorithm takes as input $params' = (params, P_o)$, and does the following:
  1) Select a random $x_p \in Z_q^*$ and compute $P_p = g^{x_p}$.
  2) Output (x_p, P_p) as the proxy signer's private-public key pair.
- Delegate($params', \lambda, m_w$): This algorithm accepts $params', \lambda$, m_w and generates the delegation as follows:
  1) Compute $y_w = H_1(m_w)$.
  2) Compute $\sigma_w = \frac{1}{\lambda + y_w} P$
  3) Output the delegation $\varpi = (m_w, \sigma_w)$.
- DVerify($params', \varpi$): This algorithm takes as input $params'$, a delegation ϖ and checks $e(\sigma_w, P_o + H_1(m_w)P) \stackrel{?}{=} g$. If the equation holds, it outputs *true*; otherwise, outputs *false*.
- PKGen($params', \varpi, x_p$): On input $(params', \varpi = (m_w, \sigma_w), x_p)$, it outputs the proxy signing key $K_p = (x_p, \sigma_w)$.
- PSign($params', m_w, P_p, K_p, m$): To sign a message m using the proxy signing key $K_p = (x_p, \sigma_w)$, the proxy signer performs the following steps.
  1) Select random $r_1, r_2 \in Z_q^*$.
  2) Compute $R = g^{r_1}, R' = g^{r_2}$, set $h = H_2(m, R, R', P_p)$.
  3) Compute $U = (x_p h + r_1)\sigma_w, v = x_p h + r_2$.
  4) Output $\sigma = (h, U, v)$ as the proxy signature.
- Verify($params', P_p, m_w, m, \sigma$): To verify the validity of above proxy signature, a verifier performs the following steps.
  1) Compute $R = e(U, P_o + H_1(m_w)P)P_p^{-h}, R' = g^v P_p^{-h}$.
  2) Verify $h \stackrel{?}{=} H_2(m, R, R', P_p)$ holds with equality. If the equation holds, output *true*; otherwise, output $false$.

The correctness of the above proxy signature scheme follows a direct verification. From Theorem 1, Theorem 2 and the underlying certificateless signature scheme's security property stated in Lemma 1 in Appendix A, we have the following security claim.

Theorem 3: The proposed proxy signature scheme is secure against both Type $\mathbb{A}$ and Type $\mathbb{B}$ adversaries.

VI. Conclusion

We investigated the relation between certificateless signatures and proxy signatures for the first time. In particular, we provided a generic construction of proxy signatures from certificateless signatures which are secure against super type I and II adversaries. The formal security analysis of our generic construction was also given. Moreover, we provided a efficient concrete instantiation of proxy signatures from a recent certificateless signature scheme.

Acknowledgments and Disclaimer

This work was supported in part by the the NSF of China under Grants 61202465, 61021004, 11061130539, 91118008, 60970114, 60970115, 91018008, 60970116, 61173154, 61003214, 61173192, 61103222; EU FP7 under Projects "DwB" and "Inter-Trust"; the Spanish Government under Projects CTV-09-634, PTA2009-2738-E, TSI-020302-2010-153, TIN2009-11689, TIN2011-27076-C03-01,CONSOLIDER INGENIO 2010 "ARES" CSD2007-0004, and TSI2007-65406-C03-01; the Government of Catalonia under Grant SGR20091135; the Shanghai NSF under Grant No. 12ZR1443500, 11ZR1411200; the Shanghai

Chen Guang Program (12CG24); the Fundamental Research Funds for the Central Universities of China; the Open Project of Shanghai Key Laboratory of Trustworthy Computing (No. 07dz22304201101) and the Shaanxi Provincial Education Department under Scientific Research Program 2010JK727. J. Domingo-Ferrer was supported in part as an ICREA-Acadèmia researcher by the Catalan Government.

REFERENCES

[1] S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In Proc. of Asiacrypt'2003, Lecture Notes in Computer Science, vol. 2894, pp. 452-473, Springer-Verlag, 2003.
[2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Proc. ACM CCS'93, pp. 62-73, 1993.
[3] A. Boldyreva, A. Palacio, B.Warinschi. Secure proxy signature schemes for delegation of signing rights. Cryptology ePrint Archive, Report 2003/096 (2003); Journal of Cryptology, 25(1), pp. 57-115, 2012.
[4] J. Lee, J. Cheon, S. Kim. An analysis of proxy signatures: Is a secure channel necessary? In Proc. CT-RSA 2003, Lecture Notes in Computer Science, vol. 2612, pp. 68-79. Springer-Verlag, 2003.
[5] B. Lee, H. Kim, K. Kim. Strong proxy signature and its application. In Proc. SCIS, pp. 603-608, 2001.
[6] Y. Dodis, J. Katz, S. Xu, M. Yung. Strong key-insulated signature schemes. In Proc. PKC 2003, Lecture Notes in Computer Science, vol. 2567, pp. 130-144, Springer-Verlag, 2002.
[7] T. Malkin, S. Obana, M. Yung. The hierarchy of key evolving signatures and a characterization of proxy signatures. In Proc. EUROCRYPT 2004, Lecture Notes in Computer Science, vol. 3027, pp. 306-322, Springer-Verlag, 2004.
[8] B. Hu, D. Wong, Z. Zhang and X. Deng. Key replacement attack against a generic construction of certificateless signature. In Proc. ACISP'2006, Lecture Notes in Computer Science, vol. 4058, pp. 235-346, Springer-Verlag, 2006.
[9] X. Huang, Y. Mu, W. Susilo, D. Wong and W. Wu. Certificateless signature revisited. In Proc. ACISP 2007, Lecture Notes in Computer Science, pp. 308-322, Springer-Verlag, 2007.
[10] X. Huang, Y. Mu, W. Susilo, F. Zhang, X. Chen. A short proxy signature Scheme: Efficient authentication in the ubiquitous World. In Proc. EUC'2005, Lecture Notes in Computer Science, 2005, vol. 3823, pp. 480-489, Springer-Verlag, 2005.
[11] M. Mambo, K. Usuda, and E. Okamoto, Proxy signature: Delegation of the power to sign messages. IEICE Trans. Fundamentals, E79-A(9), pp. 1338-1353, 1996.
[12] A. Shamir. Identity based cryptosystems and signature schemes. In Proc. Crypto'84, Lecture Notes in Computer Science, vol.196, pp. 47-53, Springer-Verlag, 1984.
[13] J.C. Schuldt, K. Matsuura, K.G. Paterson. Proxy signatures secure against proxy key exposure. In Proc. PKC'08, Lecture Notes in Computer Science, vol. 4939, pp. 141-161, Springer-Verlag, 2008.
[14] Z. Zhang, D. Wong, J. Xu and D. Feng. Certificateless public-key signature: security model and efficient construction. In Proc. ACNS'2006, Lecture Notes in Computer Science, vol. 3989, pp. 293-308, Springer-Verlag, 2006.
[15] J. Xu, Z. Zhang, and D. Feng. ID-based proxy signature using bilinear pairings. In Proc. ISPA'2005, Lecture Notes in Computer Science, vol. 3759, pp. 359-367, Springer-Verlag, 2005.
[16] F. Zhang, and K. Kim. Efficient ID-based blind signature and proxy signature from bilinear pairings. In Proc. ACISP'2003, Lecture Notes in Computer Science, vol. 2727, pp. 312-323, Springer-Verlag, 2003.
[17] L. Zhang, F. Zhang. A new provably secure certificateless signature scheme. In Proc. IEEE International Conference on Communications (ICC'2008), pp. 1685-1689, 2008.
[18] L. Zhang, F. Zhang, F. Zhang. New efficient certificateless signature scheme. In Proc. Emerging Directions in Embedded and Ubiquitous Computing (EUC'2007), Lecture Notes in Computer Science, vol. 4809, pp. 692-703, Springer-Verlag, 2007.
[19] L Zhang, F Zhang, Q Wu. Delegation of signing rights using certificateless proxy signatures. Information Sciences, vol. 184, no. 1, pp. 298-309, 2012.

APPENDIX

The signature scheme presented in Section V is based on the ZZZ certificateless signature scheme [18]. As remarked in Section III, some algorithms of a certificateless signature scheme maybe omitted in the scheme. In fact, in the ZZZ certificateless signature scheme [18], the Partial-Private-Key-Verify algorithm is omitted. For a self-contained reason, we restate the ZZZ certificateless signature scheme as follows.

- Setup(k): Given a security parameter k, the KGC chooses a cyclic additive group G_1 which is generated by P with prime order q, chooses a cyclic multiplicative group G_2 of the same order and a bilinear map $e : G_1 \times G_1 \longrightarrow G_2$, sets $g = e(P, P)$, chooses cryptographic hash functions $H_1 : \{0,1\}^* \longrightarrow Z_p^*$ and $H_2 : \{0,1\}^n \times G_2 \times G_2 \times G_2 \longrightarrow Z_p^*$, where *n* denote the bit-length of plaintexts, sets $params = (G_1, G_2, e, P, g, H_1, H_2)$. The KGC also runs $MKGen$ that accepts $params$, and, selects a random $\lambda \in Z_q^*$ as the master key, and, sets the $P_T = \lambda P$ as the master public key. The system parameter list is $params' = (G_1, G_2, e, P, g, P_T, H_1, H_2)$.
- Partial-Private-Key-Extract($params', master\text{-}key, ID$): This algorithm accepts $params'$, master-key λ and a user's identity ID, and, generates the partial private key for the user as follows:
  1) Compute $y_{ID} = H_1(ID)$.
  2) Output the partial private key $D_{ID} = \frac{1}{\lambda + y_{ID}} P$.
- Partial-Private-Key-Verify($params', ID, D_{ID}$): To verify the validity of D_{ID}, a verifier checks $e(D_{ID}, P_T + H_1(ID)P) \stackrel{?}{=} g$. If the equation holds, outputs *true*; otherwise, outputs *false*.
- Set-Secret-Value($params'$): On input $params'$, this algorithm selects a random $x \in Z_q^*$ and outputs x as the secret value.
- Set-Public-Key($params', x$): On input $params'$, a user's secret value x, this algorithm produces the user's public key $P_{ID} = g^x$.
- Set-Private-Key($params', ID, D_{ID}, x$): On input $params'$, a user's identity ID, secret value x, partial private key D_{ID}, this algorithm sets $S_{ID} = (x, D_{ID})$ as the user's private key.
- Sign($params', m, ID, P_{ID}, S_{ID}$): To sign a message m using the private key S_{ID}, the signer, whose identity is ID and the corresponding public key is P_{ID}, performs the following steps:
  1) Select random $r_1, r_2 \in Z_p^*$.
  2) Compute $R = g^{r_1}, R' = g^{r_2}$, set $h = H_2(m, R, R', P_{ID})$.
  3) Compute $U = (xh + r_1)D_{ID}, v = xh + r_2$.
  4) Output (h, U, v) as the signature on m.
- Verify($params', m, ID, P_{ID}, \sigma$): To verify a signature (U, v, w) on a message m for an identity ID and public key P_{ID}, the verifier performs the following steps:
  1) Compute $R = e(U, P_T + H_1(ID)P)P_{ID}^{-h}, R' = g^v P_{ID}^{-h}$.

2) Verify $h \stackrel{?}{=} H_2(m, R, R', P_{ID})$ holds with equality. If the equation holds, output *true*; otherwise, output $false$.

By combining two security results, *i.e.*, Theorem 1 and Theorem 2 of [18], the following claim holds.

Lemma 1: The above certificateless signature scheme is unforgeable against the super types I and II adversaries in the random oracle model.