Advances in Mathematics of Communications
Volume 14, No. 2, 2020, 177–205
doi:10.3934/amc.2020015

MALLEABILITY AND OWNERSHIP OF PROXY SIGNATURES: TOWARDS A STRONGER DEFINITION AND ITS LIMITATIONS

Sanjit Chatterjee

Department of Computer Science and Automation
Indian Institute of Science
India

Berkant Ustaoğlu

Izmir Institute of Technology
Urla, Izmir, 35430 Turkey

(Communicated by Neal Koblitz)

Abstract. Proxy signature is a cryptographic primitive that allows an entity to delegate singing rights to another entity. Noticing the ad-hoc nature of security analysis prevalent in the existing literature, Boldyreva, Palacio and Warinschi proposed a formal security model for proxy signature. We revisit their proposed security definition in the context of the most natural construction of proxy signature – delegation-by-certificate. Our analysis indicates certain limitations of their definition that arise due to malleability of proxy signature as well as signature ownership in the context of standard signature. We propose a stronger definition of proxy signature to address these issues. However, we observe that the natural reductionist security argument of the delegation-by certificate proxy signature construction under this definition seems to require a rather unnatural security property for a standard signature.

1. Introduction

Proxy signature allows an *original signer*, sometimes referred to as the *designator*, to delegate limited signing rights to another entity called the *proxy signer*. Such delegation has potential applications in diverse areas: from global distribution networks [1] to grid computing [11].

The notion of proxy signature was introduced by Mambo, Usuda and Okamoto [19]. Since then several schemes have been proposed but many of them were subsequently broken. Boldyreva, Palacio and Warinschi [7] provide a brief narrative of this initial development. They contend that there is a void in terms of rigorous formal treatment of proxy signature and the preliminary version of their work [6] is claimed to be the first one in the paradigm of provable security. The main contributions of [7] are a formal security model that, according to the authors, allows for a *rather powerful adversary* and several constructions of proxy signature which can be formally shown to be secure in the proposed security model.

In this work we revisit the security notion of proxy signature as formulated in [7]. In particular we undertake a critical analysis of the formal definition in [7] to understand to what extent it captures the potential use and threats in realistic

2010 *Mathematics Subject Classification:* Primary: 94A60; Secondary: 94A62.
Key words and phrases: Proxy signatures, malleability, owndership, multi-user security, provable security.

context. Our work is in the spirit of Koblitz and Menezes [17] who raise some pertinent questions on the problem of formulating *right* definitions in cryptography, including that of a digital signature.

Standard security definition of signature schemes [13, 14] deals with the question of unforgeability and hence, non-repudiation assuming there is only one signer in the system. There are several other works that point out limitations of the standard definition of digital signature [21, 27, 24]. It is also known that a secure signature scheme when combined with other primitives may result in overall security failure [5]. One of the concerns is that the standard definition [13, 14] does not take into account the realistic multi-user setting. A distinguishing feature of the latter is that of signature ownership. For example, an adversary can claim ownership of a signature generated by an honest signer.

Such an attack scenario, observed first in the context of a key agreement protocol, was termed as a Duplicate Signature Key Selection (DSKS) attack [5]. In a DSKS attack, given the signature σ of signer Alice on some message m under her certified public key pk, an attacker Mallory generates a certified public key pk' such that the same message-signature pair (m, σ) verifies under pk'. DSKS attacks on several signature schemes, including RSA and ECDSA as well as ways to thwart such attacks were discussed in [21]. Also see [17] for a detailed discussion on DSKS and its real-world impact. It is worth noting that these attacks show that the question of signature ownership may require additional assumptions beyond what is mandated by unforgeability of signature. For example, even though ECDSA remains unforgeable assuming the intractability of ECDLP, one can still mount a DSKS attack without violating that assumption.

Proxy signature by definition involves two signers – the designator (original signer) and a proxy who signs on behalf of the designator. So we find it pertinent to ask whether signature ownership (and DSKS type of attack) could be of any relevance in the context of proxy signature security. To delve into this question here we first recall (informally) the natural delegation-by-certificate construction of proxy signature [7]. The construction assumes the existence of a standard signature scheme. Suppose the proxy designator, Alice, wants to designate the proxy signer, Bob to sign a message $M \in \omega$ on her behalf. Alice first generates a warrant $\texttt{wrnt}$ which consists of her signature on a message containing the public key of Bob along with some associated information such as a description of the message space ω. Bob uses his signing key to generate a signature σ on an augmented message that includes the public key of Alice. The proxy signature on M is now $p\Sigma = (\texttt{wrnt}, \sigma)$. It's worth noting that by definition $p\Sigma$ consists of two components and thus opens up the possibility of malleability. Verification of proxy signature $p\Sigma$ involves checking whether $\texttt{wrnt}$ and σ are valid signatures for the respective messages under the corresponding public keys of Alice and Bob.

The formal security definition in [7] considers only one honest signer – either in the role of proxy designator or proxy signer and does not deal with the question of signature ownership or DSKS type attack. Informally speaking, security of the above delegation-by-certificate construction essentially means that an adversary cannot generate a new valid proxy signature on behalf of either the honest designator Alice or the honest proxy signer Bob.

We notice that not taking into account the question of signature ownership properly may lead to insecurity of the delegation-by-certificate construction. This point is illustrated in the paper through the example of ECDSA as standardized

in FIPS186-4 [23]. In particular, given a proxy signature $p\Sigma = (\mathtt{wrnt}, \sigma)$ for some $M \in \omega$ where *both* designator (Alice) *and* proxy signer (Bob) are *honest*, an adversary can come up with another proxy designator Mallory and a valid proxy signature $p\Sigma' = (\mathtt{wrnt}', \sigma)$ for same $M \in \omega$ where $\mathtt{wrnt}'$ is the signature of Mallory designating Bob.

Interestingly this attack can be interpreted in two different ways. From the perspective of honest proxy signer Bob, the attack amounts to a forgery of proxy signature and thus contradicts the security claim of delegation-by-certificate construction of [7]. On the other hand, for honest designator Alice this attack violates a desirable security goal of proxy signature which is not *formalized* in the security model of [7].

In order to address such security concerns we extend the definition of proxy signature by taking into account the ownership issue in the multi-user setting. Despite that we identify a general problem related to proxy signature, which stems from the definition of standard signature security. It appears that a natural definition of proxy signature security requires a rather unnatural security definition for standard signatures. Thus our work provides further evidence to the critique in [17], namely a theoretically accepted definition within the paradigm of provable security may have shortcomings in practice.

1.1. Outline of the paper. In §2 we argue for a stronger security definition of proxy signatures through some plausible attack scenarios on the delegation-by-certificate construction of [7]. The first attack scenario demonstrates how an adversarially generated public key can use the *malleability* property of proxy signatures to generate a forgery according to the security definition of [7]. The second scenario points out a deficiency in the security definition of proxy signature in [7]. Together they demonstrate a connection between malleability and insecurity of (proxy) signature. We conclude with a description of what a stronger definition should aim at to capture such scenarios.

In §3 we describe our proposed definition of proxy signature security. Since a secure proxy signature scheme assumes the existence of a secure signature scheme; we start from the security of standard signatures. We combine the standard unforgeability definition with the question of signature ownership in the multi-user setting, thereby providing a unified security definition. This is followed by a security definition of proxy signature. Discussion on its relation to the definition of [7] is in §4. We show that our proposed definition is stronger than the previous definition.

In §5 we revisit the construction of secure proxy signature scheme based on the delegation-by-certificate approach. However, the formal security argument reducing security of proxy signature to that of standard signature encounters an obstacle. It turns out that the task a proxy signature adversary faces cannot necessarily be reduced to the security of standard signature, even in the multi-user setting. We conclude with an open question: can it be the case that the new security goal is not implied by the existing definition of security for standard signature.

Notation. In the exposition the symbols $\mathtt{wrnt}, \sigma$ and $p\sigma$ denote the output of a signing algorithm $\mathcal{S}$. The symbol $p\Sigma$ is used to denote a proxy signature which may contain extra information besides a signature $p\sigma$, that is $p\Sigma = p\sigma \| \mathtt{aux}$ for some auxiliary information $\mathtt{aux}$. As usual $\|$ denotes concatenation.

2. A case for a stronger definition

The most natural construction of a proxy signature is through delegation-by-certificate. An entity, say Alice, delegates a proxy signer, say Bob, by providing a *warrant* together with a signature on the warrant. However, the naive construction does not distinguish a proxy signature and a regular signature leading to some kind of mismatch attacks [7, §4]. There is a simple solution: a standard signature on a message M is generated by first prepending 11 to M and signing this encoded message. A warrant, on the other hand, is generated by first prepending the string 00 to the warrant message to get $\mathtt{wrnt}$, while a proxy signature on M is a signature on the message $01\|M$.

However, there still remains a *malleability* issue as observed in [7, §4]. An adversary can designate an honest entity, say Bob, as a proxy signer for Alice. Given a delegation-by-certificate proxy signature of Bob on behalf of Alice, the adversary can substitute the warrant from Alice to Bob with a warrant from, say Mallory to Bob. The result is a valid proxy signature of Bob on behalf of Mallory, even though Bob never issued such a proxy signature. According to [7], the simple solution is to append Alice's public key in the message to be signed by the proxy signer Bob: instead of signing $01\|M$, Bob now signs $01\|pk_A\|M$, where pk_A is Alice's public key. With these modifications, [7] arrives at the following delegation-by-certificate construction:

[7, Construction 4.1]. Let $\mathtt{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be a signature scheme (Definition of $\mathtt{DS}$ from [7] is reproduced in Appendix A). The corresponding delegation-by-certificate proxy signature scheme is defined by the following set of algorithms $\mathtt{PS}[\mathtt{DS}] = (\mathcal{G}_1, \mathcal{K}_1, \mathcal{S}_1, \mathcal{V}_1, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$.

- The parameter- and key-generation algorithms are those of $\mathtt{DS}$: $\mathcal{G}_1 = \mathcal{G}, \mathcal{K}_1 = \mathcal{K}$.
- A standard signature for message M is obtained by prepending 11 to the message and signing the result using $\mathcal{S}$, i.e. $\mathcal{S}_1(sk, M) = \mathcal{S}(sk, 11\|M)$.
- Verification of a signature σ for message M is done by computing $\mathcal{V}_1(pk, M, \sigma) = \mathcal{V}(pk, 11\|M, \sigma)$.
- User $\mathtt{i}$, in order to designate user $\mathtt{j} \neq \mathtt{i}$ as a proxy signer for message space ω, simply sends to $\mathtt{j}$ the description ω of the message space, together with a *warrant* $\mathtt{wrnt} = \mathcal{S}(sk_{\mathtt{i}}, 00\|\mathtt{j}\|pk_{\mathtt{j}}\|\omega)$ (i.e., a signature on the message $00\|\mathtt{j}\|pk_{\mathtt{j}}\|\omega$, under the secret key of user $\mathtt{i}$). The corresponding proxy signing key of user $\mathtt{j}$ is $psk = (sk_{\mathtt{j}}, pk_{\mathtt{i}}, \mathtt{j}\|pk_{\mathtt{j}}\|\omega, \mathtt{wrnt})$.
- User $\mathtt{i}$, in order to designate itself as proxy signer for message space ω, runs $\mathcal{K}$ to obtain $(pk_{\mathtt{i}}^{'}, sk_{\mathtt{i}}^{'})$, and creates a warrant $\mathtt{wrnt} = \mathcal{S}(sk_{\mathtt{i}}, 00\|\mathtt{i}\|pk_{\mathtt{i}}^{'}\|\omega)$. The corresponding self-delegated proxy signing key of user $\mathtt{i}$ is
$$psk = (sk_{\mathtt{i}}^{'}, pk_{\mathtt{i}}, \mathtt{i}\|pk_{\mathtt{i}}^{'}\|\omega, \mathtt{wrnt}).$$
- A proxy signature by user $\mathtt{j}$ on behalf of user $\mathtt{i}$ on message $M \in \omega$ using proxy signing key $(sk, pk_{\mathtt{i}}, \mathtt{j}\|pk\|\omega, \mathtt{wrnt})$ [1], contains the identity $\mathtt{j}$ of the proxy signer, the message space description ω, the public key pk of the proxy signer, the warrant $\mathtt{wrnt}$ (a signature for $00\|\mathtt{j}\|pk\|\omega$ under $sk_{\mathtt{i}}$) and a signature for $01\|pk_{\mathtt{i}}\|M$ under sk. Formally,
$$\mathcal{PS}\left((sk, pk_{\mathtt{i}}, \mathtt{j}\|pk\|\omega, \mathtt{wrnt}), M\right) = \left(\mathtt{j}, \omega, pk, \mathtt{wrnt}, \mathcal{S}(sk, 01\|pk_{\mathtt{i}}\|M)\right).$$

[1] Here $(pk, sk) = (pk_{\mathtt{j}}, sk_{\mathtt{j}})$ if $\mathtt{i} \neq \mathtt{j}$ and $(pk, sk) = (pk_{\mathtt{i}}^{'}, sk_{\mathtt{i}}^{'})$ if $\mathtt{i} = \mathtt{j}$.

If $M \notin \omega$ the signing algorithm returns $\perp$.

- Proxy signature verification is defined via

$$\mathcal{PV}(pk', M, \perp) = 0$$

and

$$\mathcal{PV}\left(pk', M, (\mathtt{j}, \omega, pk, \mathtt{wrnt}, \sigma)\right) = \mathcal{V}(pk', 00\|\mathtt{j}\|pk\|\omega, \mathtt{wrnt}) \wedge \mathcal{V}(pk, 01\|pk'\|M, \sigma) \wedge (M \in \omega).$$

- The identification algorithm is defined as $\mathcal{ID}(j, \omega, pk, \mathtt{wrnt}, \sigma) = j$.

To formulate a strong security definition, Boldyreva et al. [7] claimed to have considered an extreme scenario where only one user (denoted as user `1`) is honest. The paper summarizes the goal of an adversary against the proxy signature scheme as follows.

G1 forgery of a standard signature: signature σ produced by the adversary is valid under the public key of user `1`;

G2 forgery of a proxy signature by user `1` on behalf of user $j \neq$ `1`: the proxy signature belongs to user j even though user `1` was not designated by user j, or if there was a designation then the message was not queried to the corresponding proxy signing oracle;

G3 forgery of a proxy signature by user `1` on behalf of user `1`: the proxy signature belongs to user `1` even though user `1` did not self-designate, or if there was such a designation then the message was not queried to the corresponding proxy signing oracle;

G4 forgery of a proxy signature of user $i \neq$ `1` on behalf of user `1`: user i was not designated by user `1` to sign M.

Based on these goals a formal security model [7, Definition 3.2] is proposed and shown that the delegation-by-certificate [7, Construction 4.1] is secure in that model. Essentially, if an adversary can achieve any of the above goals then that amounts to forgery in the underlying signature scheme.

An Attack Scenario. We have already recalled how [7] addressed the problem of malleability in proxy signature by including the designator's public key as part of the message signed by the proxy signer (see [7, Construction 4.1] reproduced above). However, this measure does not necessarily prevent forgery by exploiting malleability of a proxy signature by user `1` on behalf of user i as we show next. In this attack, the adversary is given a proxy signature $p\Sigma = (\mathtt{wrnt}, \sigma)$ by proxy signer user `1` on behalf of designator user `i`. The adversary can generate a valid proxy signature $p\Sigma' = (\mathtt{wrnt}', \sigma)$ for the same message where $\mathtt{wrnt}'$ is the warrant from some user `o` created by the adversary with public key *same* as that of user `i`. Thus in this attack scenario the adversary can achieve goal G2 described above.

The attack uses the Elliptic Curve Digital Signature (ECDSA) as standardized in FIPS186-4 [23, §6]. See Appendix B.1 for a description of the ECDSA signing and verification algorithm. In [23] the domain parameters for ECDSA are $(q, FR, a, b, \{, \text{domain\_parameter\_seed}\}, G, n, h)$ where q, FR correspond to the underlying field and (a, b) describe the elliptic curve. A (private-public) key pair is defined as (d, Q) for $d \in [1, n-1]$ and $Q = dG$ for a generating point G on the elliptic curve as specified in the standard. This is in line with other standards such as [2, §2.3.5], where a point on the curve is represented as an "OCTET STRING", or [22], where the (private-public) key pairs are also an integer d and an elliptic curve point

Q, or [8, §3.2]. The important point to note is that, as per these standards, a public key is simply an element of an elliptic curve group.

Let the public key of user $\mathtt{1}$ be $pk_{\mathtt{1}}$, where $pk_{\mathtt{1}} = sk_{\mathtt{1}}G_{\mathtt{1}}$. The associated parameters were generated by user $\mathtt{1}$ as recommended in FIPS186-4 [23], with parameters

$$(q_{\mathtt{1}}, FR_{\mathtt{1}}, a_{\mathtt{1}}, b_{\mathtt{1}}, \{, \text{domain\_parameter\_seed}_{\mathtt{1}}\}, G_{\mathtt{1}}, n_{\mathtt{1}}, h_{\mathtt{1}}).$$

User $\mathtt{i}$ has public key $pk_{\mathtt{i}}$ where $pk_{\mathtt{i}} = sk_{\mathtt{i}}G_{\mathtt{i}}$ with associated parameters

$$(q_{\mathtt{i}}, FR_{\mathtt{i}}, a_{\mathtt{i}}, b_{\mathtt{i}}, \{, \text{domain\_parameter\_seed}_{\mathtt{i}}\}, G_{\mathtt{i}}, n_{\mathtt{i}}, h_{\mathtt{i}}).$$

Suppose user $\mathtt{i}$ designates user $\mathtt{1}$ via warrant $\mathtt{wrnt}_{\mathtt{i}\to\mathtt{1}}$ on message space ω and user $\mathtt{1}$ issues a proxy signature on some $m \in \omega$ on behalf of user $\mathtt{i}$ as $p\Sigma = (\mathtt{1}, \omega, pk_{\mathtt{1}}, \mathtt{wrnt}_{\mathtt{i}\to\mathtt{1}}, \sigma)$. According to [7, Construction 4.1], proxy signature verification is

$$\begin{aligned}&\mathcal{PV}\left(pk_{\mathtt{i}}, M, (\mathtt{1}, \omega, pk_{\mathtt{1}}, \mathtt{wrnt}_{\mathtt{i}\to\mathtt{1}}, \sigma)\right)\\ &\quad = \mathcal{V}\left(pk_{\mathtt{i}}, 00\|\mathtt{1}\|pk_{\mathtt{1}}\|\omega, \mathtt{wrnt}_{\mathtt{i}\to\mathtt{1}}\right) \wedge \mathcal{V}\left(pk_{\mathtt{1}}, 01\|pk_{\mathtt{i}}\|M, \sigma\right) \wedge (M \in \omega).\end{aligned}$$

To mount the attack, the adversary creates a warrant $\mathtt{wrnt}_{\mathtt{o}\to\mathtt{1}}$ on behalf of user $\mathtt{o} \neq \mathtt{i}$ under a message space ω' with $M \in \omega'$ and with public key $pk_{\mathtt{o}} = pk_{\mathtt{i}}$. Then by substituting the warrant issued by user $\mathtt{i}$ with the one issued by user $\mathtt{o}$, the adversary creates a new proxy signature on M as $p\Sigma' = (\mathtt{1}, \omega', pk_{\mathtt{1}}, \mathtt{wrnt}_{\mathtt{o}\to\mathtt{1}}, \sigma)$. It is easy to verify that $p\Sigma'$ will be accepted as a valid proxy signature issued by user $\mathtt{1}$ on behalf of user $\mathtt{o}$, even though user $\mathtt{1}$ was not designated by user $\mathtt{o}$ thereby defeating Goal G2 outlined above.

Note that in the security model of [7] there is only one honest user, namely, user $\mathtt{1}$. Thus assuming user $\mathtt{i}$ to be malicious, user $\mathtt{o}$ could obtain the private key corresponding to $pk_{\mathtt{o}}$ from user $\mathtt{i}$ and create the warrant $\mathtt{wrnt}_{\mathtt{o}\to\mathtt{1}}$. This is a valid attack since [7] does not explicitly require that different users have different public keys, in fact the adversary is implicitly allowed to register any (valid) key. However, let's additionally assume that the adversary has no access to the private information of user $\mathtt{i}$ i.e., both the designator and proxy signer are deemed to be honest. In other words, we consider the multi-user setting. The above attack can nevertheless be mounted by crafting a suitable certificate.

As per FIPS186-4 [23], a user is allowed to generate public parameters of her choice. To mount the attack user $\mathtt{o}$ creates a malicious certificate with public key $pk_{\mathtt{o}} = pk_{\mathtt{i}}$ and parameters identical to those used by user $\mathtt{i}$ except for the generating point $G_{\mathtt{i}}$. Given the parameters of user $\mathtt{i}$,

$$(q_{\mathtt{i}}, FR_{\mathtt{i}}, a_{\mathtt{i}}, b_{\mathtt{i}}, \{, \text{domain\_parameter\_seed}_{\mathtt{i}}\}, G_{\mathtt{i}}, n_{\mathtt{i}}, h_{\mathtt{i}}),$$

user $\mathtt{o}$ selects the same curve but with generating point $G_{\mathtt{o}} = sk_{\mathtt{o}}^{-1}pk_{\mathtt{i}}$ where $sk_{\mathtt{o}}^{-1}$ is a random integer modulo $n_{\mathtt{i}}$. This is possible, see [23, §6.1.1], as it is not mandatory to exhibit a proof that G is generated at random. In Appendix B.2.2 we have described the above attack on a concrete toy example. Note that this attack is closely related to but different from DSKS attacks on ECDSA reported in the literature (see, for example, [17, §2.2.4]).

A Second Scenario. We discuss another plausible attack scenario which is not covered by any of the goals G1–G4 in [7]. Suppose an honest user $\mathtt{1}$ located in the Far East wants to enter an auction taking place in Europe. To avoid late night bidding user $\mathtt{1}$ grants a warrant to a proxy (user $\mathtt{i}$) to enter the auction on behalf of user $\mathtt{1}$. Suppose user $\mathtt{i}$ makes a winning bid. A little later an entity (user $\mathtt{o}$)

located in the west coast of America learns about the results and wants to claim the item auctioned.

Let the warrant issued by user 1 to user i be denoted by $\mathtt{wrnt}_{1\to i}$ on the message space ω. As per [7, Construction 4.1], the proxy signature on message M – the bid by user i on user 1's behalf – is

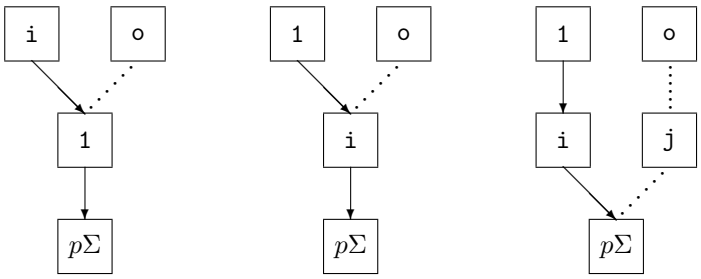
$$p\Sigma = (\mathtt{i}, \omega, pk_{\mathtt{i}}, \mathtt{wrnt}_{1\to i}, p\sigma) \text{ where } p\sigma = \mathcal{S}(sk_{\mathtt{i}}, 01\|pk_{\mathtt{1}}\|M).$$

The proxy verification algorithm takes as input the designator's public key $pk_{\mathtt{1}}$, the message M and a proxy signature. Now user o crafts a certificate with public key $pk_{\mathtt{o}} = pk_{\mathtt{1}}$ and private key $sk_{\mathtt{o}}$ as discussed in the attack scenario above; then generates a new warrant $\mathtt{wrnt}_{\mathtt{o}\to\mathtt{i}}$ on message space ω designating user i as proxy signer. With that

$$p\Sigma' = (\mathtt{i}, \omega, pk_{\mathtt{i}}, \mathtt{wrnt}_{\mathtt{o}\to\mathtt{i}}, p\sigma)\,, \text{ where as before } p\sigma = \mathcal{S}(sk_{\mathtt{i}}, 01\|pk_{\mathtt{1}}\|M),$$

is a 'new' proxy signature on the same bid M that verifies under the public key $pk_{\mathtt{o}}$ of user o.

Some similarities notwithstanding, there is a fundamental difference between the two attack scenarios. In the first case, the honest user is the *proxy signer* whereas in the second the honest user is the *designator*. We illustrate this point in Figure 1. Figure 1(a) depicts the forgery attack in the first scenario where $p\Sigma$ is a proxy signature of user o even though honest user 1 had not issued such signature (or perhaps not even received authorization to sign on user o's behalf); (b) is the ownership attack in the second scenario (and its generalization is in (c)). Given a proxy signature that belongs to honest designator user 1 and using the inherent malleability of $p\Sigma$, user o is able to generate a valid proxy signature $p\Sigma'$. Clearly this is not a forgery of the honest designator user 1. Since there is no assumption about the honesty of proxy signer user i, neither can this be construed as a forgery of user i. Since [7, Definition 3.2] considers only proxy signature forgery and not ownership, attacks depicted in Figure 1(b) and (c) are not covered in that security definition.



(a) (b) (c)

Figure 1. Different attack scenarios

3. Proxy signature: A candidate definition

The above discussion suggests that the security definition of proxy signature needs to be augmented: not only should an adversary be unable to forge signatures, but she should also be unable to misappropriate signatures of honest users, whether signed via a proxy or not.

The original security model of digital signature [13, 14], formulated as existential unforgeability under chosen message attack (EUF-CMA), considers a single signer. The security of signature schemes in the multi-user setting has been further investigated by several researchers (for example, [21, 24, 15]). These works demonstrate the need to formalize the notion of signature ownership as part of the security definition in the multi-user setting. Pornin and Stern [24], in particular, proposed three different notions of ownership of signature. We recall their definition of universal exclusive ownership from [24], which is strongest of the three.

Definition 3.1 (Universal exclusive ownership)**.** [24, Definition 3] Given a public key pk and t pairs (M_i, σ_i) of message-signature that verify under the public key pk, it is infeasible to create a public key $pk^{'} \neq pk$ and a matching private key $sk^{'}$ such that at least one message-signature pair (M_i', σ'), with $\sigma' = \sigma_i$ for some $1 \leq i \leq t$ verifies under $pk^{'}$. Further, there is no restriction that $M_i' = M_i$.

3.1. SIGNATURE SECURITY: A COMBINED DEFINITION. Rather than considering multiple security aspects separately (existential unforgeability versus universal exclusive ownership; single user versus multi-user setting) we prefer to use a single security definition that encompasses all the above aspects. This will allow us to focus only on the relation between the new primitives and definitions we state later in the context of proxy signature and to simplify security arguments.

Definition 3.2 (EU-UEO-CMA)**.** We say that a signature scheme in the multi-user setting is existentially unforgeable providing universal exclusive ownership under adaptive chosen message key attack *EU-UEO-CMA* if an adversary who can adaptively request t public keys with associated certificates, for each is given a signing oracle that can adaptively provide at most $t^{\mathtt{i}}$ signatures for the corresponding $\mathtt{cert}$, and adaptively request private keys corresponding to $\mathtt{cert}$, cannot output $(\mathtt{cert_i}, M, \sigma)$ or $[(\mathtt{cert_i}, M, \sigma), (\mathtt{cert_o}, M', \sigma)]$ such that the triplet $(\mathtt{cert_i}, M, \sigma)$ is a valid signature under $\mathtt{cert_i}$, the certificate $\mathtt{cert_i}$ is requested by the adversary, the adversary did not query for the corresponding private key and one of the following holds:

1. either the message M was not queried to the oracle with $\mathtt{cert_i}$;
2. or (M, σ) is a response from the oracle with $\mathtt{cert_i}$ and (M', σ) is a valid signature under $\mathtt{cert_o}$ (here $\mathtt{cert_o} \neq \mathtt{cert_i}$).

Note that in our definition we use a certificate $\mathtt{cert}$ instead of a public key. The original UEO definition deals only with public keys implicitly assuming one certificate per public key. Thus a relevant issue is whether an honest user can possess two certificates with the *same* public key. If UEO is defined via certificates, then there is a trivial UEO attack if we allow the same public key under, say, two different certificates. In this work we will assume honest users generate novel key pairs for every new certificate. The certificates may share public parameters such as underlying elliptic curve or field, as well as information such as the identifier of a party. In other words, there is a one-to-one correspondence between public keys and the certificates of honest users.

It is straightforward to show that EU-UEO-CMA security is implied by EU-CMA and UEO security. For the sake of completeness we state this in the following theorem (see Appendix C.1 for the proof).

Theorem 3.3. *Let* $\mathtt{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ *be a signature scheme that is both EU-CMA and UEO secure. Then it is also EU-UEO-CMA secure.*

3.2. PROXY SIGNATURE. In the following, rather than talking about users we will talk about certificates and assume that each certificate is unique; instead of saying that user $\mathtt{i}$ designates user $\mathtt{j}$ to sign certain message we say that a certificate $\mathtt{cert}_\mathtt{i}$ designates another certificate $\mathtt{cert}_\mathtt{j}$ to sign messages. The entity owning the certificate $\mathtt{cert}_\mathtt{j}$ is responsible for maintaining the corresponding private key.

Definition 3.4 (proxy signature)**.** A *proxy signature scheme* is described by a tuple $\mathrm{PS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV})$, where the constituent algorithms run in polynomial time, $\mathrm{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is a digital signature scheme, and the other components are defined as follows:

- $(\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is a signature scheme;
- $(\mathcal{D}, \mathcal{P})$ is a pair of interactive randomized algorithms forming the (two-party) *proxy-designation protocol*. The input to each algorithm includes two *certificates* $\mathtt{cert}_\mathtt{i}$, $\mathtt{cert}_\mathtt{j}$ for the *designator* $\mathtt{i}$ and the *proxy signer* $\mathtt{j}$, respectively. $\mathcal{D}$ also takes as input the secret key $sk_{\mathtt{cert}_\mathtt{i}}$ corresponding to public key listed in $\mathtt{cert}_\mathtt{i}$, and a message space descriptor ω for which user $\mathtt{i}$ utilizing $\mathtt{cert}_\mathtt{i}$ wants to delegate signing rights to user $\mathtt{j}$ owning $\mathtt{cert}_\mathtt{j}$. $\mathcal{P}$ also takes as input the secret key $sk_{\mathtt{cert}_\mathtt{j}}$ corresponding to the public key listed in $\mathtt{cert}_\mathtt{j}$. As a result of the interaction, the expected local output of $\mathcal{P}$ is *spk*, a proxy signing key that user $\mathtt{j}$ uses to produce proxy signature on behalf of user $\mathtt{i}$, for messages in ω. $\mathcal{D}$ has no local output. We write $skp \leftarrow [\mathcal{D}(\mathtt{cert}_\mathtt{i}, sk_{\mathtt{cert}_\mathtt{i}}, \mathtt{cert}_\mathtt{j}, \omega), \mathcal{P}(\mathtt{cert}_\mathtt{j}, sk_{\mathtt{cert}_\mathtt{j}}, \mathtt{cert}_\mathtt{i})]$ for the result of this interaction; the designation is identified by $\mathtt{cert}_\mathtt{i} \overset{w}{\to} \mathtt{cert}_\mathtt{j}$.
- $\mathcal{PS}$ is the (possibly) randomized *proxy signing* algorithm. It takes input a chain $\mathtt{cert}_\mathtt{i} \overset{w}{\to} \mathtt{cert}_\mathtt{j}$, a proxy signing key *skp* and a message $M \in \{0,1\}^*$, and outputs a proxy signature $p\Sigma \in \{0,1\}^* \cup \{\perp\}$ along with some, possibly empty, auxiliary information $\mathtt{aux}$;
- $\mathcal{PV}$ is the deterministic *proxy verification* algorithm. It takes input a chain $\mathtt{cert}_\mathtt{i} \overset{w}{\to} \mathtt{cert}_\mathtt{j}$ a message $M \in \{0,1\}^*$, a proxy signature $p\Sigma$ and auxiliary input $\mathtt{aux}$, and outputs 0 or 1. In the latter case, we say that $p\Sigma$ is a *valid* proxy signature for M relative to $\mathtt{cert}_\mathtt{i} \overset{w}{\to} \mathtt{cert}_\mathtt{j}$.

Correctness. We require that (i) the signature scheme is correct and (ii) for any message space $\omega \subseteq \{0,1\}^*$ and for all users $\mathtt{i}, \mathtt{j} \in \mathbb{N}$, if *skp* is a proxy signing key of user $\mathtt{j}$ on behalf of user $\mathtt{i}$ for message space ω, i.e.,

$$skp \leftarrow [\mathcal{D}(\mathtt{cert}_\mathtt{i}, sk_{\mathtt{cert}_\mathtt{i}}, \mathtt{cert}_\mathtt{j}, \omega), \mathcal{P}(\mathtt{cert}_\mathtt{j}, sk_{\mathtt{cert}_\mathtt{j}}, \mathtt{cert}_\mathtt{i})],$$

then for every $M \in \omega$, with probability one

$$\mathcal{PV}\big(\mathtt{cert}_\mathtt{i} \overset{w}{\to} \mathtt{cert}_\mathtt{j}, M, \mathcal{PS}(\mathtt{cert}_\mathtt{i} \overset{w}{\to} \mathtt{cert}_\mathtt{j}, skp, M)\big) = 1.$$

For notational simplicity, we will write $\mathtt{cert}_\mathtt{i} \to \mathtt{cert}_\mathtt{j}$ in place of $\mathtt{cert}_\mathtt{i} \overset{w}{\to} \mathtt{cert}_\mathtt{j}$ when w is clear from the context. Observe that, unlike [7], our definition does not have the identification algorithm. We have dispensed with it as it seems the algorithm does not have any role in the adversary's security goal.

Remark 1. Unlike the definition in [7], the output of $\mathcal{PS}$ is a (proxy) signature along with some auxiliary information. Similarly, the input to $\mathcal{PV}$ is a (proxy) signature along with auxiliary input. This, however, does not modify the algorithms in any fundamental way. Indeed by redefining $p\Sigma$ as the concatenation of $p\Sigma$ with $\mathtt{aux}$ the definition can dispense with $\mathtt{aux}$. However, for clarity of presentation we

prefer this separation. For the particular construction [7, Construction 4.1] the original notation is

$$p\Sigma = (\mathtt{j}, \omega, pk, \mathtt{wrnt}, \mathcal{S}(sk, 01\|pk_{\mathtt{i}}\|M))$$

whereas with our notation,

$$\mathtt{aux} = (\mathtt{j}, \omega, pk, \mathtt{wrnt}), \qquad p\Sigma = \mathcal{S}(sk, 01\|pk_{\mathtt{i}}\|M).$$

In our opinion, this division is natural since multiple proxy signatures can share the values in aux. Furthermore, the mapping between the two notation is a bijection thus previous security arguments apply with the new notation as well.

3.3. SECURITY MODEL. Our model distinguishes between users/entities and certificates. A user may own more than one certificate but for an honest user there is a one-to-one correspondence between each public key and certificate. Users that are not adversary controlled follow the protocol specifications. When interacting with other entities, the interaction is based on information provided in a certificate. If two distinct entities possess and use identical certificates the related responsibility lies with the issuing authority.

Honest users interact with *valid certificates* only – such as those meeting the criteria in [9, 28]. Honest users own certificates for which they generate public parameters and keys according to specifications and it is possible that parameters except public keys are shared across multiple certificates.

The adversary can also create certificates, but is not obliged to follow specifications when creating public parameters or keys. However, to interact with any user the adversary must have a valid certificate. Any adversary-created certificate is called *adversary controlled* or *adversarial*. Due to various reasons users may expose their private keys to adversaries. Such action is allowed in our model and the corresponding certificates are labeled *discredited*. Together, adversary controlled and discredited certificates are called *compromised*. Our model implies no forgery guarantee for compromised certificates, but they play an important role in the context of signature ownership.

3.3.1. *Adversarial queries.* Here we describe the adversarial environment. The adversarial queries are a *generalization* of the queries allowed in [7], which had a single honest user, here the number of honest users is polynomially bounded (in the security parameter κ). For each honest user the adversary is allowed similar type of queries as in [7].

Let PS $= (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV})$ be the proxy signature scheme, $\mathcal{A}$ is an adversary against PS who runs in time polynomial in $\kappa \in \mathbb{N}$. We associate to PS, $\mathcal{A}$ and κ an experiment $\mathbf{Exp}^{\mathrm{ps-uf}}_{\mathrm{PS},\mathcal{A}}(\kappa)$. A counter HU (honest users) is set to one. $\mathcal{A}$ can make the following queries, in any order.

new i: $\mathcal{A}$ is given access to a new honest user indexed by $\mathtt{i} = HU$ and HU is increased by one. Associated with the honest user there is a (per user) counter $\mathtt{c}_{\mathtt{i}}$ set initially to zero ($\mathtt{c}_{\mathtt{i}}$ will index the certificates generated for user i). The experiment initializes empty lists $\mathbf{DU}_{\mathtt{i}}$, $\mathbf{PU}_{\mathtt{i}}$, $\mathbf{CS}_{\mathtt{i}}$ and $\mathbf{CC}_{\mathtt{i}}$. The set $\mathbf{DU}_{\mathtt{i}}$ stores the certificates designated by the user i (together with the message spaces for which they are designated and with which certificate the designation was done). The set $\mathbf{PU}_{\mathtt{i}}$ keeps the designations for which the user can proxy sign messages along with signing keys; $\mathbf{CS}_{\mathtt{i}}$ keeps track of the set of messages (ω) for which the adversary can produce proxy signature by user

$\mathtt{i}$ using compromised proxy signing keys. And lastly $\mathbf{CC}_{\mathtt{i}}$ keeps track of the certificates for which the adversary obtained the corresponding private key.

new $(\mathtt{cert}_{\mathtt{i}}, \text{info}, \mathtt{i})$**:** If info does not contain domain parameters, then these parameters *params* are generated by running $\mathcal{G}$ on input 1^{κ}. A new key pair is generated via $(sk_{\mathtt{i}}^{c_{\mathtt{i}}}, pk_{\mathtt{i}}^{c_{\mathtt{i}}}) \leftarrow \mathcal{K}(params)$ Then a new certificate $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$ incorporating *params*, $pk_{\mathtt{i}}^{c_{\mathtt{i}}}$ is generated for user $\mathtt{i}$; the counter $c_{\mathtt{i}}$ is incremented by one. The certificate becomes available to all users; the private key corresponding to the public key listed in the certificate is available only to user $\mathtt{i}$. In particular the adversary does not know the corresponding private key. We say user $\mathtt{i}$ *owns the certificate* $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$. Any other information contained in info is included in fields allowed in $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$. For example, the adversary can require that a certain id string (such as email address) associated with entity $\mathtt{i}$ be used in the generated certificate.

register $\mathtt{cert}_{\mathtt{o}}$**:** $\mathcal{A}$ can request to register a certificate $\mathtt{cert}_{\mathtt{o}}$ that becomes available to all users. The certificate is adversary controlled (i.e. compromised).

$\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$ **designates** $\mathtt{cert}_{\mathtt{o}}$**:** $\mathcal{A}$ can request to interact with user $\mathtt{i}$, whereby $\mathtt{i}$ runs proxy delegation algorithm $\mathcal{D}(\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}, sk_{\mathtt{i}}^{c_{\mathtt{i}}}, \mathtt{cert}_{\mathtt{o}}, \omega)$, for some certificated $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$ owned by $\mathtt{i}$, some registered certificate $\mathtt{cert}_{\mathtt{o}}$ and some message space ω (chosen by $\mathcal{A}$). In the interaction $\mathcal{A}$ plays the role of user $\mathtt{o}$ running $\mathcal{P}(\mathtt{cert}_{\mathtt{o}}, sk_{\mathtt{o}}, \mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}})$. After a successful run $[\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}, \omega, \mathtt{cert}_{\mathtt{o}}]$ is appended to $\mathbf{DU}_{\mathtt{i}}$.

$\mathtt{cert}_{\mathtt{o}}$ **designates** $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$**:** $\mathcal{A}$ can request to interact with user $\mathtt{i}$, whereby $\mathtt{i}$ runs the proxy delegation algorithm $\mathcal{P}(\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}, sk_{\mathtt{i}}^{c_{\mathtt{i}}}, \mathtt{cert}_{\mathtt{o}})$ for some certificated $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$ owned by $\mathtt{i}$, some registered certificate $\mathtt{cert}_{\mathtt{o}}$ and a message space ω chosen by the adversary. In the interaction $\mathcal{A}$ plays the role of user $\mathtt{o}$ running $\mathcal{D}(\mathtt{cert}_{\mathtt{o}}, sk_{\mathtt{o}}, \mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}, \omega)$. If *skp* is the resulting proxy signing key, then entry $[skp, \mathtt{cert}_{\mathtt{o}}, \omega, \mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}]$ is appended to the array $\mathbf{PU}_{\mathtt{i}}$. $\mathcal{A}$ does not have direct access to the first element of an entry in the array $\mathbf{PU}_{\mathtt{i}}$.

$\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$ **designates** $\mathtt{cert}_{\mathtt{j}}^{c_{\mathtt{j}}}$**:** $\mathcal{A}$ can request that user $\mathtt{i}$ runs the delegation protocol with user $\mathtt{j}$, where $\mathtt{i} = \mathtt{j}$ is allowed for some message space ω selected by the adversary. $\mathcal{A}$ is given the transcript of the interaction. Let $skp \leftarrow [\mathcal{D}(\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}, sk_{\mathtt{i}}^{c_{\mathtt{i}}}, \mathtt{cert}_{\mathtt{j}}^{c_{\mathtt{j}}}, \omega), \mathcal{P}(\mathtt{cert}_{\mathtt{j}}^{c_{\mathtt{j}}}, sk_{\mathtt{j}}^{c_{\mathtt{j}}}, \mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}})]$ be the resulting proxy signing key, then $[skp, \mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}, \omega, \mathtt{cert}_{\mathtt{j}}^{c_{\mathtt{j}}}]$ is appended to $\mathbf{PU}_{\mathtt{j}}$ and $[\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}, \omega, \mathtt{cert}_{\mathtt{j}}^{c_{\mathtt{j}}}]$ is appended to $\mathbf{DU}_{\mathtt{i}}$. $\mathcal{A}$ does not have direct access to the first element of an entry in the array $\mathbf{PU}_{\mathtt{j}}$.

exposure of user $\mathtt{i}$**'s** l**th proxy signing key:** $\mathcal{A}$ can request to see $\mathbf{PU}_{\mathtt{i}}[l]$ for some $l \in \mathbb{N}$. If $\mathbf{PU}_{\mathtt{i}}[l]$ contains a proxy signing key then *spk* is returned to $\mathcal{A}$ and $\mathbf{CS}_{\mathtt{i}}$ is set to $\mathbf{CS}_{\mathtt{i}} \cup \omega$. Otherwise, $\perp$ is returned to $\mathcal{A}$.

exposure of certificate $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$ **owned by user** $\mathtt{i}$**:** $\mathcal{A}$ can request the private key associated with $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$. In that case $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$ becomes compromised and $[sk_{\mathtt{i}}^{c_{\mathtt{i}}}, \mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}]$ is appended to $\mathbf{CC}_{\mathtt{i}}$.

standard signature for certificate $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$**:** $\mathcal{A}$ can query oracle $\mathcal{O}_{\mathcal{S}}(\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}, \cdot)$ with a message M and obtain a standard signature σ for M by user $\mathtt{i}$ signed using certificate $\mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}$.

proxy signature of user $\mathtt{i}$ **using the** l**th proxy signing key:** $\mathcal{A}$ can make a query $(\mathtt{i}, l, M)$, where $\mathtt{i}$ is an honest user, $l \in \mathbb{N}$ and $M \in \{0,1\}^*$, to oracle $\mathcal{O}_{\mathcal{PS}}((\mathtt{cert}_{\mathtt{j}}^{c_{\mathtt{j}}} \rightarrow \mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}), \cdot)$. If $\mathbf{PU}_{\mathtt{i}}[l] = [spk, \mathtt{cert}_{\mathtt{j}}^{c_{\mathtt{j}}}, \omega, \mathtt{cert}_{\mathtt{i}}^{c_{\mathtt{i}}}]$ and $M \in \omega$, we say the query is *valid* and the oracle returns the proxy signature

$p\Sigma = \mathcal{PS}((\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}} \rightarrow \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}), spk, M)$. Otherwise, we say the query is *invalid* and the oracle returns $\perp$.

3.3.2. *Adversarial goal.* Eventually, $\mathcal{A}$ outputs one of the followings.

1. forgery of the form $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, \sigma)$, where $\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}$ is not corrupted, M was not queried to oracle $\mathcal{O}_{\mathcal{S}}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \cdot)$ and $\mathcal{V}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, \sigma) = 1$, then return 1. [forgery of a standard signature];
2. forgery of the form $(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}} \rightarrow \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, p\Sigma \| \mathtt{aux})$, such that $\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}$ is not corrupted and for all $l \in \mathbb{N}$ with $\mathcal{PU}_{\mathtt{i}}[l] = (spk, \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \omega, \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}})$, no valid $(\mathtt{i}, l, M)$ query was issued to $\mathcal{O}_{\mathcal{PS}}((\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}} \rightarrow \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}), \cdot)$, and $\mathcal{PV}(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}} \rightarrow \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, p\Sigma \| \mathtt{aux}) = 1$ then return 1. [forgery of a proxy signature by an honest user $\mathtt{i}$];
3. forgery of the form $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \rightarrow \mathtt{cert}_{\mathtt{o}}, M, p\Sigma \| \mathtt{aux})$, such that for each message space ω for which $[\mathtt{cert}_{\mathtt{o}}, \omega, \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}] \in \mathbf{DU}_{\mathtt{i}}$ it holds that $M \notin \omega$ and it also holds that $\mathcal{PV}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \rightarrow \mathtt{cert}_{\mathtt{o}}, M, p\Sigma \| \mathtt{aux}) = 1$ then return 1. [forgery of a proxy signature of user $\mathtt{o} \neq \mathtt{i}$ on behalf of user $\mathtt{i}$; user $\mathtt{o}$ was not designated by user $\mathtt{i}$ to sign M];
4. one of the following (see Remark 2 below):
   (a) two chains $\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \rightarrow \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}$ and $\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}} \rightarrow \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}$, for which at least one of $\{\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}, \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}\}$ is not compromised, messages (M, M') and a tuple $(p\Sigma, \mathtt{aux}, \mathtt{aux}')$ such that
   $$\begin{aligned} \mathcal{PV}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \rightarrow \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M, p\Sigma \| \mathtt{aux}) &= 1 \\ \mathcal{PV}(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}} \rightarrow \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}, M', p\Sigma \| \mathtt{aux}') &= 1 \end{aligned}$$
   (b) or a chain $\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \rightarrow \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}$ and a certificate $\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}$ for which at least one of $\{\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}} \mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}\}$ is not compromised, messages (M, M') and a tuple $(p\Sigma, \mathtt{aux})$, such that
   $$\begin{aligned} \mathcal{PV}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \rightarrow \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M, p\Sigma \| \mathtt{aux}) &= 1 \\ \mathcal{V}(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}, M', p\Sigma) &= 1 \end{aligned}$$
   (c) or two certificates $\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}$ and $\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}$ for which at least one is not compromised, two messages (M, M') and a signature σ such that
   $$\begin{aligned} \mathcal{V}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, \sigma) &= 1 \\ \mathcal{V}(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M', \sigma) &= 1 \end{aligned}$$
   then return 1.
5. output $\perp$, return 0.

We define the *advantage* of adversary $\mathcal{A}$ as

$$\mathbf{Adv}_{\mathrm{PS},\mathcal{A}}^{\mathrm{ps-uf}}(\kappa) = \Pr[\mathbf{Exp}_{\mathrm{PS},\mathcal{A}}^{\mathrm{ps-uf}}(\kappa) = 1].$$

We say that PS is a *secure proxy signature scheme* if the function $\mathbf{Adv}_{\mathrm{PS},\mathcal{A}}^{\mathrm{ps-uf}}(\cdot)$ is negligible for all adversaries $\mathcal{A}$ of time complexity polynomial in the security parameter κ.

Remark 2. Goal 4 does not have any restriction that $\mathtt{aux} \neq \mathtt{aux}'$ or $M \neq M'$. This goal pertains to the question of (proxy) signature ownership. Our definition is motivated by *universal exclusive ownership* [24, Definition 3] and the similar idea of *message and key substitution attack* outlined in [21, §2.2 item 4]. The

goal implicitly assumes that the message associated with the certificate that is not compromised has been queried to the corresponding signing oracle e.g., if $\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}$ in 4(c) is not compromised then query $\mathcal{O}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M)$ was issued. It is easy to see that otherwise $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, \sigma)$ is a valid signature forgery as in Goal 1. One can weaken the goal of adversary to allow for the variants like *conservative exclusive ownership* [24, Definition 1] and *destructive exclusive ownership* [24, Definition 2], or [21, Definition 4]. Note that the adversary is considered successful even when s/he explicitly requests a signature for an uncorrupted certificate and then creates a different chain/certificate under which the same signature verifies.

4. Relation to previous definition

As with any new definition, a natural question is how the proposed definition compares with the existing one. The following theorem shows that our proxy signature definition is no weaker than the one in [7].

Theorem 4.1. *A proxy signature scheme secure in the sense of §3.3 is secure in the sense of* [7, Definition 3.2].

Proof. Given an adversary $\mathcal{A}$ in the sense of [7, Definition 3.2] we construct an algorithm $\mathcal{R}$ that transforms $\mathcal{A}$ into an adversary $\mathcal{M}$ in the sense of §3.3. In the remainder of the argument `aux` is empty and omitted from the exposition.

The algorithm $\mathcal{R}$ is initiated with its inputs and starts by issuing **new** `1` and **new** $(\mathtt{cert}_1, \text{info}, 1)$ where info is empty; as a result $\mathtt{cert}_1^1 = \mathtt{cert}_1$. Let pk_1 be the resulting public key. The algorithm $\mathcal{A}$ is initiated with pk_1 and $\mathcal{R}$ responds to queries by $\mathcal{A}$ in the following way:

- **registers** pk_i**:** $\mathcal{R}$ issues **register** $\mathtt{cert}_{\mathtt{o}}$ with public key pk_i (we assume a signature scheme that incorporates validation procedures that allow for valid certificate generation);
- `1` **designates** i**:** $\mathcal{R}$ issues $\mathtt{cert}_{\mathtt{i}}^1$ **designates** $\mathtt{cert}_{\mathtt{i}}$;
- i **designates** `1`**:** $\mathcal{R}$ requests that $\mathtt{cert}_{\mathtt{i}}$ **designates** $\mathtt{cert}_1^1$;
- `1` **designates** `1`**:** $\mathcal{R}$ issue **new** $\mathtt{cert}_1^{\mathtt{c_1}}$ with public key $pk_1^{\mathtt{c_1}}$; increments $\mathtt{c}_1$ and requests that $\mathtt{cert}_1^1$ **designates** $\mathtt{cert}_1^{\mathtt{c_1}}$.
- **exposure of self delegation:** for the lth proxy signing key, $\mathcal{R}$ repeats the query asking for the lth proxy signing key of user `1`. We note that if the proxy signing key is simply the key in the lth certificate of user `1` the certificate is considered compromised but as far as the environment of $\mathcal{A}$ is concerned the existence or non-existence of the certificate is independent from obtaining the proxy signing key; the response is forwarded to $\mathcal{A}$;
- **standard signature by** `1`**:** $\mathcal{R}$ queries $\mathcal{O}_{\mathcal{S}}(\mathtt{cert}_1^1, \cdot)$ and forwards the response to $\mathcal{A}$;
- **proxy signature of user** `1` **on behalf of user** `i`**:** using the lth proxy signing key – $\mathcal{R}$ queries $\mathcal{O}_{\mathcal{PS}}(\mathtt{cert}_{\mathtt{i}} \to \mathtt{cert}_1^1, \cdot)$ with message and information provided in $\mathcal{A}$'s query; e.g., $\mathtt{cert}_{\mathtt{i}}$ has the public key provided by $\mathcal{A}$, which could in particular be a public key generated for user `1` during self delegations, and the remaining information such as the message is included in the oracle query. The response is forwarded to $\mathcal{A}$;

The simulation of $\mathcal{A}$'s environment is therefore perfect, and depending on $\mathcal{A}$'s output $\mathcal{R}$ outputs

forgery of standard signature: if $\mathcal{A}$ successfully outputs (M, Σ), where M was not queried to oracle $\mathcal{O}_{\mathcal{S}}(sk_1, \cdot)$ and $\mathcal{V}(pk_1, M, \Sigma) = 1$, then $\mathcal{R}$ outputs $(\mathtt{cert}_1^1, M, \Sigma)$ as a standard signature forgery and $\mathcal{R}$ is successful;

forgery of a proxy signature by user $\mathtt{1}$ on behalf of user $\mathtt{i} \neq \mathtt{1}$: if $\mathcal{A}$ successfully outputs $(M, p\Sigma, pk_{\mathtt{i}})$, for some $\mathtt{i} \in \{2 \dots n\}$, where

1. $\mathcal{ID}(p\Sigma) = 1$,
2. $\mathcal{PV}(pk_{\mathtt{i}}, M, p\Sigma) = 1$ and
3. for $l \in \mathbb{N}$ no valid query $(\mathtt{1}, l, M)$ was made to $\mathcal{O}_{\mathcal{PS}}((\mathbf{spk}_u)_{u\in[n]}, \cdot)$,

then $\mathcal{R}$ outputs $(\mathtt{cert}_{\mathtt{i}} \to \mathtt{cert}_1^1, M, p\Sigma)$, where $pk_{\mathtt{i}}$ is embedded in $\mathtt{cert}_{\mathtt{i}}$, as a valid proxy signature forgery of user $\mathtt{1}$ and $\mathcal{R}$ is successful;

forgery of a proxy signature by user $\mathtt{1}$ on behalf of user $\mathtt{1}$: if $\mathcal{A}$ successfully outputs $(M, p\Sigma, pk_1)$, where

1. $\mathcal{ID}(p\Sigma) = 1$,
2. $\mathcal{PV}(pk_1, M, p\Sigma) = 1$,
3. $M \notin \mathbf{CS}_1$, and
4. no valid query $(\mathtt{1}, l, M)$, for $l \in \mathbb{N}$, was made to $\mathcal{O}_{\mathcal{PS}}((\mathbf{spk}_u)_{u\in[n]}, \cdot)$,

then $\mathcal{R}$ outputs $(\mathtt{cert}_1^{\mathtt{c}_1} \to \mathtt{cert}_1^1, M, p\Sigma)$, here $\mathtt{cert}_1^{\mathtt{c}_1}$ has the public key corresponding to sk_u; the output is a valid proxy signature forgery and $\mathcal{R}$ is successful;

forgery of a proxy signature on behalf of $\mathtt{1}$ by $\mathtt{i} \neq \mathtt{1}$ where $\mathtt{1}$ did not designate $\mathtt{i}$:
if $\mathcal{A}$ outputs $(M, p\Sigma, pk_1)$, where

1. $\mathcal{PV}(pk_1, M, p\Sigma) = 1$ and
2. for each ω such that $(\mathcal{ID}(p\Sigma), \omega) \in \mathbf{DU}_1$ it holds that $M \notin \omega$,

then $\mathcal{R}$ outputs $(\mathtt{cert}_1^1 \to \mathtt{cert}_{\mathtt{i}}, M, p\sigma)$, where $\mathtt{cert}_{\mathtt{i}}$ contains the public key corresponding to the identity $\mathcal{ID}(p\Sigma)$, which is a valid proxy signature forgery and $\mathcal{R}$ is successful.

no success: in all other cases $\mathcal{R}$ terminates unsuccessfully.

So if $\mathcal{A}$ is successful then $\mathcal{R}$ is also successful; equivalently if the proxy signature is secure in the sense of §3.3 it is secure in the sense of [7, Definition 3.2]. □

Our proposed security definition of proxy signature also encompasses signature security definitions from [24] (see also [21, §2.2]). In particular, given a message-signature pair under the public key of an honest user, the definition implies that there is no efficient algorithm to create a new public-private key pair under which the same message-signature pair will validate. This feature of preventing public key substitution attack means our definition addresses signature ownership issue for both signature and proxy signature setting in a unified way and we get the following result.

Theorem 4.2. *If a proxy signature is secure in the sense of §3.3 then the signature scheme within the proxy signature is secure in the sense of Definition 3.2.*

Proof. As with the argument for Theorem 4.1, we proceed by showing that the contrapositive holds. In particular, an EU-UEO adversary $\mathcal{A}$ against the signature via a reduction $\mathcal{R}$ can be transformed into an adversary against the proxy signature. The algorithm $\mathcal{R}$ is initiated with its inputs, access to $\mathcal{A}$ and starts by issuing **new** $\mathtt{i}$ and **new** $(\mathtt{cert}_{\mathtt{i}}, \mathrm{info}, \mathtt{i})$ for $\mathtt{i} = 1 \dots n$ where info is empty for each index $\mathtt{i}$ and n is the number of public keys $\mathcal{A}$ takes as input. The algorithm $\mathcal{A}$ is initiated and $\mathcal{R}$ responds to ith certificate query by returning the ith certificate. For

every message M queried by $\mathcal{A}$ for signature, the algorithm $\mathcal{R}$ queries the corresponding $\mathcal{O}(\mathtt{cert_i}, M)$ and forwards the response to $\mathcal{A}$. When $\mathcal{A}$ outputs $\mathtt{cert_o}$ the algorithm $\mathcal{R}$ registers a new certificate $\mathtt{cert_o}$. The output of $\mathcal{A}$ is

1. a message signature $(\mathtt{cert_i}, M, \sigma)$ such that no $\mathcal{O}(\mathtt{cert_i}, M)$ query was issued and $\mathcal{V}(\mathtt{cert_i}, M, \sigma) = 1$ in which case $\mathcal{R}$ outputs $(\mathtt{cert_i}, M, \sigma)$ and successfully achieves Goal 1 in §3.3;
2. or $[(\mathtt{cert_i}, M, \sigma), (\mathtt{cert_o}, M', \sigma)]$ where $\mathcal{O}(\mathtt{cert_i}, M)$ was issued and
$$\mathcal{V}(\mathtt{cert_i}, M, \sigma) = 1 \qquad \text{and} \qquad \mathcal{V}(\mathtt{cert_o}, M', \sigma) = 1.$$
In this case $\mathcal{R}$ outputs $[(\mathtt{cert_i}, M, \sigma), (\mathtt{cert_o}, M', \sigma)]$ and succeeds in achieving point 4(c) in adversary goal described in §3.3.

In either case the environment for $\mathcal{A}$ is simulated perfectly and if $\mathcal{A}$ is successful in polynomial time then so is $\mathcal{R}$, which completes the argument. □

An immediate corollary using Theorem 3.3 is that the existence of a proxy signature scheme secure in the sense of §3.3 implies the existence of a signature scheme that is UEO-secure.

Finally, the following claim establishes that our security definition is stronger that of [7].

Lemma 4.3. *There exists a proxy signature scheme which is secure in the sense of* [7, Definition 3.2] *but insecure in the security model of §3.3.*

The claim is established through a delegation-by-certificate proxy signature construction. Note that we have already shown an attack on [7, Construction 4.1] in their own security model. However, by replacing the public key used in [7, Construction 4.1] by the corresponding certificate one can easily show that this construction is secure in the sense of [7, Definition 3.2]. This modification is easy to achieve because we have assumed that there is a one-to-one correspondence between public keys and certificates for honest users.

Given this generic construction of a proxy signature we now build a separating example, which closely follows [15, Example 1] that we recall first.

Let $\mathrm{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be any EU-CMA secure signature scheme. Define $\mathrm{DS}' = (\mathcal{G}', \mathcal{K}', \mathcal{S}', \mathcal{V}')$ as the *cheat* variant of DS.

1. $\mathcal{G}'$ is a randomized algorithm that on input 1^κ outputs a set of domain parameters *params* by running $\mathcal{G}$. The algorithm may output some additional information I that can be used to verify some properties of the output.
2. $\mathcal{K}'$ is a randomized algorithm which on input a set of domain parameters *params* outputs with overwhelming probability a (private-public) key pair (sk, pk) associated with domain parameters *params* by running $\mathcal{K}$. However, with negligible probability $\mathcal{K}'$ outputs a special key pair $(\mathtt{Cheat}, \mathtt{Key})$ valid for all domain parameters;
3. $\mathcal{S}'$ upon called with key `Key` outputs a special signature `ValidOnlyForCheat`, in all other cases $\mathcal{S}$ is executed;
4. $\mathcal{V}'$ first checks if the verification is against the key `Cheat`. In this case the signature is accepted without any further checks. In all other cases, $\mathcal{V}$ is run and its output returned.

Since the key generation algorithm outputs `Cheat` with a negligible probability, the signature scheme $\mathcal{DS}'$ is EU-CMA secure. Using [7, Construction 4.1] we obtain a proxy signature scheme secure in the sense of [7, Definition 3.2]. However, this

scheme will be insecure in the new definition of §3.3. In particular an adversary can easily achieve Goal 4(a) (and Goal 4(b)) of §3.3.2 as described below.

Following [7, Construction 4.1], we can define a proxy signature scheme as follows: $\mathrm{PS}' = ((\mathcal{G}', \mathcal{K}', \mathcal{S}', \mathcal{V}')(\mathcal{D}', \mathcal{P}'), \mathcal{PS}', \mathcal{PV}')$

- $\mathcal{G}'$ returns the output of $\mathcal{G}$;
- $\mathcal{K}'$ is a randomized algorithm which on input a set of domain parameters *params* outputs with overwhelming probability a (private-public) key pair (sk, pk) associated with domain parameters *params* by running $\mathcal{K}$. However, with negligible probability $\mathcal{K}'$ outputs one of the special key pairs $(\mathtt{Cheat_s}, \mathtt{Key_s})$ and $(\mathtt{Cheat_p}, \mathtt{Key_p})$ valid for all domain parameters. The former is used for ordinary signature and the latter for proxy signature.
- $\mathcal{S}'$ upon called with key $\mathtt{Key_s}$ outputs a special signature $\mathtt{11ValidOnlyFor Cheat_s}$, upon called with key $\mathtt{Key_p}$ outputs a special signature $\mathtt{11ValidOnly ForCheat_p}$, in all other cases $\mathcal{S}$ is executed;
- $\mathcal{V}'$ first checks if the verification is against the key $\mathtt{Cheat_s}$ or $\mathtt{Cheat_p}$ then the signature is accepted without any further checks. In all other case $\mathcal{V}$ is run and its output returned.
- $(\mathcal{D}', \mathcal{P}')$ is equivalent to $(\mathcal{D}, \mathcal{P})$ of [7, Construction 4.1] except that when signing with either $\mathtt{Cheat_s}$ or $\mathtt{Cheat_p}$ in the warrant issuing step the output is $\mathtt{00ValidOnlyForCheat_s}$ and $\mathtt{00ValidOnlyForCheat_p}$ respectively.
- PS' runs PS of [7, Construction 4.1] except when signing with $\mathtt{Cheat_s}$ (resp. $\mathtt{Cheat_p}$) in the warrant issuing step. When signing with $\mathtt{Cheat_s}$ (resp. $\mathtt{Cheat_p}$), the output will be $\mathtt{10ValidOnlyForCheat_s}$ (resp. $\mathtt{10ValidOnlyForCheat_p}$).
- PV' : if the verification key is $\mathtt{Key_s}$ and the proxy key is $\mathtt{Key_p}$ then accept the proxy signature without further verifications, else run PV of [7, Construction 4.1] and return its output.

The above scheme is identical to [7, Construction 4.1] except with negligible probability of generating either of the cheat keys. Hence the construction is secure in the sense of [7]. On the other hand given any proxy signature of user $\mathtt{1}$, the adversary can succeed in achieving Goal 4(a) by setting the keys $\mathtt{Cheat_s}$ and $\mathtt{Cheat_p}$ for the designator and proxy respectively and claim any proxy signature belonging to this set of users. The Goal 4(b) can also be achieved in a similar fashion.

5. Revisiting security of delegation-by-certificate construction

With the appropriate modification in the security definition as discussed above, one may expect to finally establish that delegation-by-certificate leads to a provably secure proxy signature construction. We discuss the security of such a generic construction which is an extension of [7, Construction 4.1].

5.1. Proxy signature construction. Let $\mathrm{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be a signature scheme. We define the corresponding delegation-by-certificate proxy signature scheme as follows $\mathrm{PS[DS]} = (\mathcal{G}_p, \mathcal{K}_p, \mathcal{S}_p, \mathcal{V}_p, , \mathcal{PS}, \mathcal{PV})$. The construction assumes the existence of a cryptographic hash function $\mathcal{H}$ with appropriate range and domain.

- The parameter- and key-generation algorithms are those of DS: $\mathcal{G}_p = \mathcal{G}$, $\mathcal{K}_p = \mathcal{K}$. The resulting public parameters and public keys are encoded in a certificate $\mathtt{cert}$.
- A standard signature for message M under certificate $\mathtt{cert}$ with corresponding private key $sk_{\mathtt{cert}}$ is $\sigma = \mathcal{S}_p(sk_{\mathtt{cert}}, M) = \mathcal{S}(sk_{\mathtt{cert}}, \mathcal{H}(\mathtt{cert}) \| M)$.

- Verification of a signature for message M under information provided by $\mathtt{cert}$ is done by computing $\mathcal{V}_p(pk_{\mathtt{cert}}, M, \sigma) = \mathcal{V}(pk_{\mathtt{cert}}, \mathcal{H}(\mathtt{cert})\|M, \sigma)$.
- Proxy delegation $(\mathcal{D}, \mathcal{P})$ works as follows. User $\mathtt{i}$ with certificate $\mathtt{cert}_{c_\mathtt{i}}$, in order to designate user $\mathtt{j}$ with certificate $\mathtt{cert}_{c_\mathtt{j}} \neq \mathtt{cert}_{c_\mathtt{i}}$ as a proxy signer for messages in message space ω, sends to $\mathtt{j}$ the description ω of the message space, together with a *warrant* which is a signature on the message $\mathcal{H}(\mathtt{cert}_{c_\mathtt{i}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{i}}\|\mathtt{cert}_{c_\mathtt{j}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{j}})\|\omega$ under the secret key corresponding to $\mathtt{cert}_{c_\mathtt{i}}$. In other words,

  $$\mathtt{wrnt} = \mathcal{S}(sk_{\mathtt{cert}_{c_\mathtt{i}}}, \mathcal{H}(\mathtt{cert}_{c_\mathtt{i}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{i}}\|\mathtt{cert}_{c_\mathtt{j}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{j}})\|\omega).$$

  The corresponding proxy signing key of user $\mathtt{j}$ is

  $$skp = (sk_{\mathtt{cert}_{c_\mathtt{j}}}, \mathtt{cert}_{c_\mathtt{i}}\|\mathtt{cert}_{c_\mathtt{j}}\|\omega, \mathtt{wrnt}).$$

- A proxy signature by user $\mathtt{j}$ on behalf of user $\mathtt{i}$ on message $M \in \omega$ using proxy signing key $skp = (sk_{\mathtt{cert}_{c_\mathtt{j}}}, \mathtt{cert}_{c_\mathtt{i}}\|\mathtt{cert}_{c_\mathtt{j}}\|\omega, \mathtt{wrnt})$, is

  $$\mathcal{PS}\left(\mathtt{cert}_{c_\mathtt{i}} \to \mathtt{cert}_{c_\mathtt{j}}, skp, M\right) = p\Sigma\|\mathtt{aux}$$

  where

  $$\begin{aligned} p\Sigma &= \mathcal{S}(sk_{\mathtt{cert}_{c_\mathtt{j}}}, \mathcal{H}(\mathtt{cert}_{c_\mathtt{i}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{i}}\|\mathtt{cert}_{c_\mathtt{j}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{j}})\|M); \\ \mathtt{aux} &= \mathtt{cert}_{c_\mathtt{i}}\|\mathtt{cert}_{c_\mathtt{j}}\|\omega, \mathtt{wrnt} \end{aligned}$$

  assuming $M \in \omega$, else the signing algorithm returns $\perp$.
- Proxy signature verification $\mathcal{PV}(\mathtt{cert}_{c_\mathtt{i}} \to \mathtt{cert}_{c_\mathtt{j}}, M, p\Sigma\|\mathtt{aux})$ is defined via $\mathcal{PV}(\mathtt{cert}_{c_\mathtt{i}} \to \mathtt{cert}_{c_\mathtt{j}}, M, \perp) = 0$ and

  $$\begin{aligned} &\mathcal{PV}(\mathtt{cert}_{c_\mathtt{i}} \to \mathtt{cert}_{c_\mathtt{j}}, M, p\Sigma\|\mathtt{aux}) = \\ &\qquad \mathcal{V}\left(pk_{\mathtt{cert}_{c_\mathtt{i}}}, \mathcal{H}(\mathtt{cert}_{c_\mathtt{i}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{i}}\|\mathtt{cert}_{c_\mathtt{j}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{j}})\|\omega, \mathtt{wrnt}\right) \\ &\quad \wedge \quad \mathcal{V}\left(pk_{\mathtt{cert}_{c_\mathtt{j}}}, \mathcal{H}(\mathtt{cert}_{c_\mathtt{i}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{i}}\|\mathtt{cert}_{c_\mathtt{j}})\|\mathcal{H}(\mathtt{cert}_{c_\mathtt{j}})\|M, p\Sigma\right) \\ &\quad \wedge \quad (M \in \omega). \end{aligned}$$

User $\mathtt{i}$ using certificate $\mathtt{cert}_{c_\mathtt{i}}$ is not allowed to designate the same certificate $\mathtt{cert}_{c_\mathtt{i}}$ – in line with [7] – avoiding circular self-delegations.

5.2. ON SECURITY. We want to formally establish that the above construction is secure in the security model of §3.3 provided the underlying signature scheme is EU-UEO-CMA-secure (Definition 3.2). The natural approach is to give a reduction. In other words, show that if a polynomial time adversary against the proxy signature scheme can achieve one of the four goals as formulated in §3.3.2 with some non-negligible advantage then one can construct a polynomial time adversary which breaks the EU-UEO-CMA security of the underlying signature scheme with some non-negligible advantage.

Such a reductionist argument turns out to be quite straightforward when there is a (proxy) signature forgery. When the goal of the (proxy) signature adversary is one of the Goals (1), (2), (3) of §3.3.2, the reduction can easily produce a forgery for the underlying signature scheme. We provide a sketch towards such a reduction in Appendix C.2.

However, things take a somewhat unexpected turn when one considers security in terms of (proxy) signature ownership, as defined in Goal (4) of §3.3.2. As an illustration, consider Goal 4(a) from §3.3.2. In this case the proxy signature adversary is considered successful if the following event occurs:

The adversary returns the following: $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \rightarrow \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M, p\Sigma \| \mathtt{aux})$ and $(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}} \rightarrow \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}, M', p\Sigma \| \mathtt{aux}')$ such that at least one of $\{\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}, \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}\}$ is not compromised and

$$\begin{aligned} \mathcal{PV}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \rightarrow \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M, p\Sigma \| \mathtt{aux}) &= 1, \\ \mathcal{PV}(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}} \rightarrow \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}, M', p\Sigma \| \mathtt{aux}') &= 1 \end{aligned}$$

hold.

In other words, the proxy signature adversary breaks the "ownership" property of the underlying proxy-signature scheme. In such an eventuality we expect to show that the signature adversary can break the UEO-security of the underlying signature scheme. However, it appears that to formally establish such a claim we need either an additional assumption on the proxy signer or to change the exclusive ownership definition for signature schemes.

Modifying the assumption: One way to circumvent the problem is to additionally assume that one of the two proxy certificates $\{\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}\}$ is not compromised. We elaborated this approach in the reduction argument of Appendix C.2. In that case the reduction $\mathcal{R}$ returns the UEO forgery

$$[(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \tilde{M}, p\Sigma), (\mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}, \tilde{M}', p\Sigma)],$$

where

$$\begin{aligned} \tilde{M} &= \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \| \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}) \| M \\ \tilde{M}' &= \mathcal{H}(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}} \| \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}) \| M'. \end{aligned}$$

Thus, from the perspective of an honest designator, for the desired reductionist security claim to hold s/he needs the additional assumption that the proxy is *honest*. Arguably such a restriction is extremely difficult to ensure in the real world. Furthermore, any natural security definition of proxy signature needs to consider scenarios where a party tries to produce a proxy signature even when not authorized to do so. Hence we do not think that assuming honest proxy is a realistic assumption.

Modifying the definition: The other alternative seems to be a modification in the corresponding definition of exclusive ownership for signature schemes. If both the proxy signers in the above mentioned scenario of Goal 4(a) are controlled by the adversary then in terms of the underlying signature scheme the adversarial goal is the following: given a signature scheme $(\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, produce two public keys (pk_1, pk_2), two messages (m_1, m_2) that when signed with the two public keys (pk_1 and pk_2) will result in the same signature (σ). Obviously this goal is *different* from the adversarial goal in exclusive universal ownership definition [24] where the adversary is *given* one of the public keys.

In the UEO-security game, the target public key and associated parameters are generated as specified by the signature scheme whereas in the above scenario the adversary may craft parameters and both the public keys in a malicious fashion. In [27] the signature scheme includes an algorithm that validates domain parameters along with standard verification of signature. However, as noted in §2, validating public parameters does not prevent dishonest users from claiming (proxy) signature of honest users. Thus adding a parameter verification algorithm as suggested in [27] does not necessarily bridge the gap.

In fact if we restrict our attention to ordinary signature schemes then such problem looks somewhat artificial: an adversary who may not honestly follow parameter

generation, key generation and signature generation, produces two public keys with associated public parameters, two (perhaps distinct) messages which result in the same signature that validates under two different public keys.

The work of Bohli, Röhrich and Steinwandt [15] does consider the case of malicious signers. In [15, Definition 3] they consider a *single* message-signature pair and *two* public keys generated by the adversary. Whereas the adversary's goal in the above reduction is to craft *two* public keys and *two* (distinct) messages that verify under the same signature. Both the problems are trivial for ECDSA [23] (see Appendix B.3 for an example of the latter). From the perspective of an honest user, her keys are not involved so no liability can be placed on the user. Thus there appears to be little practical motivation to modify the security definition of standard signature to incorporate the above *attack* scenario.

Remark 3. Instead of modifying either the signature security definition or the assumption about honest proxy, one may attempt to find a middle way. Suppose we only require that a single proxy is incorporated in Goal 4 of proxy signature security definition. In this scenario, both the designators delegate to the same proxy signer. The adversarial goal now would be to create one public key, two distinct messages, one signature such that the two messages verify under the same signature public key pairs. The problem sounds more restrictive than the previous case where the adversary can pick two distinct public keys. Nevertheless, reduction to UEO still seems unlikely due to the same issue: in UEO the adversary is *given* the target public keys whereas here the adversary *crafts* the public parameters and keys.

Remark 4. Independently, the issue of producing the private keys (or prove their possession) corresponding to public keys (called *weak* and *strong* goals in [15]) can be incorporated in all of the above problem variants. Nevertheless, the difference in the problem definition that lies in being *given* a public key and *crafting* a public key still remains. As already mentioned, there appears to be little motivation to consider adversarial actions that are not related to any public keys associated with honest users. The fact that the adversary may or may not produce the corresponding private key has little relevance to the practicality of these problems.

Perhaps a stronger requirement on the adversary than just a proof-of-possession of private keys may work: for example a proof that the adversary did indeed follow the protocol specifications of $\mathcal{G}$ and $\mathcal{K}$, so that *crafting* public parameters and keys is equivalent to being *given* public keys. Unfortunately, for standardized, state-of-the-art signature schemes like ECDSA such assumptions may well be false.

Various works [21, 15, 27, 24] suggest that definition of signature security in the multi-user setting is not yet completely understood. Consequently, we may lack the "right" definition of security of signature in the multi-user setting. Given that proxy signature by definition requires a multi-user setting, it may well be the case that existing signature security definitions are not sufficient to craft an appropriate proxy signature security definition. In fact obtaining the right proxy signature definition may be as challenging as defining a secure signature in the multi-user setting. Furthermore, there is no guarantee that a secure signature scheme will be sufficient to obtain a secure proxy signature scheme. It is plausible to assume that, along with right definition of signature in a multi user setting, a necessary requirement for proxy signature would be a "right" definition of delegation of trust.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their elaborate and insightful comments which helped us in improving the presentation as well as address some of the shortcomings.

APPENDIX A. DEFINITIONS

For completeness here we recall definition and security model of (proxy) signature from [7]. Note that we have slightly modified the standard signature definition by introducing certificate corresponding to a public key.

Definition A.1. [Signature scheme] For a security parameter κ a signature scheme is a tuple of algorithms $(\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ where

1. $\mathcal{G}$ is a randomized algorithm that on input 1^κ outputs a set of domain parameters $params$. The algorithm may output some additional information I that can be used to verify some properties of the output;
2. $\mathcal{K}$ is a randomized algorithm which on input a set of domain parameters $params$ outputs a private-public key pair (sk, pk) associated with domain parameters $params$. The values $(pk, params)$ can be unambiguously embedded in and extracted from a description called *certificate* and denoted by $\mathtt{cert}$; the algorithm may output some additional information that can be used to verify that the parameters were honestly generated;
3. $\mathcal{S}$ is a randomized algorithm that on input a message M, a private key sk with associated certificate containing system parameters $params$ and a public key pk, outputs a digital signature $\sigma \in \{0,1\}^* \cup \perp$;
4. $\mathcal{V}$ is a deterministic algorithm that on input a $\mathtt{cert}$ – a certificate incorporating $params$ and a public key pk, a message M and a digital signature σ, outputs 1 or 0.
$$\mathcal{V}(\mathtt{cert}, M, \sigma) = \begin{cases} 1 \\ 0 \end{cases}$$
If $\mathcal{V}(\mathtt{cert}, M, \sigma) = 1$ we say that the signature σ is valid relative to $\mathtt{cert}$.

Signature correctness. A signature scheme is correct if for a certificate $\mathtt{cert}$ containing $(pk, params)$, where sk and pk are associated with $params$, we have that $\mathcal{V}(\mathtt{cert}, M, \mathcal{S}(\mathtt{cert}, sk, M)) = 1$.

Definition A.2 (proxy signatures[7])**.** A *proxy signature scheme* is described by a tuple $\mathrm{PS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$, where the constituents are polynomial time algorithms, $\mathrm{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is a digital signature scheme, and

- $(\mathcal{D}, \mathcal{P})$ is a pair of interactive randomized algorithms forming the (two-party) *proxy-designation protocol*. The input to each algorithm includes the public key $pk_{\mathtt{i}}$ of the designator $\mathtt{i}$, and the public key $pk_{\mathtt{j}}$ of proxy signer. $\mathcal{D}$ also takes as input the secret key $sk_{\mathtt{i}}$ corresponding to $pk_{\mathtt{i}}$, the identity $\mathtt{j}$ and a message space descriptor ω. $\mathcal{P}$ also takes as input the secret key $sk_{\mathtt{j}}$. As a result, the expected local output of $\mathcal{P}$ is a proxy signing key spk used to produce proxy signatures on behalf of user $\mathtt{i}$ for messages in ω. $\mathcal{D}$ has no local output.
- $\mathcal{PS}$ is the (possibly) randomized *proxy signing* algorithm that takes as input a proxy signing key skp and a message $M \in \{0,1\}^*$, and outputs a proxy signature $p\Sigma \in \{0,1\}^* \cup \{\perp\}$.

- $\mathcal{PV}$ is the deterministic *proxy verification* algorithm. It takes input a public key $pk_{\mathtt{i}}$, a message $M \in \{0,1\}^*$, a proxy signature $p\Sigma$ and outputs 0 or 1.
- $\mathcal{ID}$ is an *identification algorithm* that on input a valid proxy signature $p\Sigma$, outputs an identity or $\perp$ in case of an error.

Correctness. For any message space $\omega \subseteq \{0,1\}^*$ and for all users $\mathtt{i}, \mathtt{j} \in \mathbb{N}$, if skp is a proxy signing key of user $\mathtt{j}$ on behalf of user $\mathtt{i}$ for message space ω, then for every $M \in \omega$, with probability one $\mathcal{PV}\big(pk_{\mathtt{i}}, M, \mathcal{PS}(skp, M)\big) = 1$ and $\mathcal{ID}(\mathcal{PS}(spk, M)) = \mathtt{j}$.

Security Model. The adversarial environment in [7] involves a single honest user. Let $\mathrm{PS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV})$ be a proxy signature scheme, $\mathcal{A}$ an adversary and $\kappa \in \mathbb{N}$. Associated to PS, $\mathcal{A}$ and κ is an experiment $\mathbf{Exp}^{\mathrm{ps-uf}}_{\mathrm{PS},\mathcal{A}}(\kappa)$. Initially, system parameters are generated using $\mathcal{G}$ and a public-private key pair $(sk_{\mathtt{1}}, pk_{\mathtt{1}})$ is generated for user $\mathtt{1}$ using $\mathcal{K}$. Empty set **DU** and **CS** are created along with an empty array **SD**; here **DU** is the set of users designated by $\mathtt{1}$, **CS** is the set of users that designated user $\mathtt{1}$, and **SD** is the array that stores the self designated proxy signing keys and corresponding message spaces. The value n is initialized to one.

Adversary $\mathcal{A}$, given $pk_{\mathtt{1}}$, can make the following queries (in any order) polynomially in κ number of times.

- $\mathtt{i}$ **register** $pk_{\mathtt{i}}$**:** $\mathcal{A}$ registers a key $pk_{\mathtt{i}}$ for user $\mathtt{i} = n + 1$. The key is stored and the counter n is incremented. An empty array $skp_{\mathtt{i}}$ is created that will store the proxy signing keys of user $\mathtt{1}$ on behalf of user $\mathtt{i}$, along with the message space descriptor.
- $\mathtt{i}$ **designates** $\mathtt{1}$**:** On $\mathcal{A}$'s choice of message space ω for some $\mathtt{i} \geq 2$ the adversary interacts with $\mathtt{1}$ that executes $\mathcal{D}(pk_{\mathtt{1}}, sk_{\mathtt{1}}, \mathtt{i}, pk_{\mathtt{i}}, \omega)$. In the interaction $\mathcal{A}$ plays the role of user $\mathtt{i}$ running the appropriate $\mathcal{P}$. After a successful run $[\mathtt{i}, \omega]$ is appended to $\mathbf{DU}_{\mathtt{i}}$.
- $\mathtt{1}$ **designates** $\mathtt{i}$**:** $\mathcal{A}$ interacts with user $\mathtt{1}$, whereby $\mathtt{i}$ runs $\mathcal{P}(pk_{\mathtt{i}}, sk_{\mathtt{i}}, pk_{\mathtt{1}})$, for some $\mathtt{i} \geq 2$. In the interaction $\mathcal{A}$ plays the role of user $\mathtt{i}$ for a message space ω of the adversary's choice. If skp is the resulting proxy signing key, then entry $[skp, \omega]$ is appended to the array $spk_{\mathtt{i}}$. $\mathcal{A}$ does not have direct access to $spk_{\mathtt{i}}$.
- $\mathtt{1}$ **designates** $\mathtt{1}$**:** $\mathcal{A}$ interacts with user $\mathtt{1}$, whereby $\mathtt{1}$ designates itself. As a result $\mathcal{A}$ is given the transcript output and $[spk, \omega]$ is appended to $spk_{\mathtt{1}}$, where spk is the resulting proxy signing key and ω is the message space description.
- **exposure of user** $\mathtt{1}$**'s** l**th proxy signing key during self delegation:** $\mathcal{A}$ can request the self delegation proxy signing key stored at $spk_{\mathtt{1}}[l]$. In this case the corresponding key is given to $\mathcal{A}$ and **CS** is set to $\mathbf{CS} \cup \omega$. If no key exists at $spk_{\mathtt{1}}[l]$, $\perp$ is returned to $\mathcal{A}$.
- **standard signature by** $\mathtt{1}$**:** $\mathcal{A}$ can query oracle $\mathcal{O}_{\mathcal{S}}(pk_{\mathtt{1}}, M)$ with a message M and obtain a standard signature σ for M by user $\mathtt{1}$.
- **proxy signature of user** $\mathtt{1}$ **on behalf of user** $\mathtt{i}$**:** (using the lth proxy signing key) $\mathcal{A}$ can make a query $(\mathtt{i}, l, M)$. As a result $\mathcal{A}$ obtains the proxy signature on message M using proxy signing key stored at $spk_{\mathtt{i}}[l]$, assuming there is such key and M belongs the corresponding message space ω. Otherwise the query is *invalid* and the response is $\perp$.

Adversarial goal. Eventually, $\mathcal{A}$ outputs a forgery (M, Σ) or $(M, p\Sigma, pk)$. The output of the experiment is

1. if the forgery is (M, Σ), M was not queried to the standard signature oracle and $\mathcal{V}(pk_{\mathtt{1}}, M, \Sigma) = 1$, then return 1; [forgery of a standard signature];

2. if the forgery is of the form $(M, p\Sigma, pk_{\mathtt{i}})$ where $\mathcal{PV}(pk_{\mathtt{i}}, M, p\Sigma) = 1, \mathcal{ID}(p\Sigma) = 1$ and no valid $(\mathtt{i}, l, M)$ query for all l was made to the proxy signing oracle return 1; [forgery of a proxy signature by user $\mathtt{1}$ on behalf of user $i \neq \mathtt{1}$];
3. if the forgery is of the form $(M, p\Sigma, pk_{\mathtt{1}})$ where $\mathcal{PV}(pk_{\mathtt{1}}, M, p\Sigma) = 1, \mathcal{ID}(p\Sigma) = 1$ and no valid $(\mathtt{1}, l, M)$ query for all l was made to the proxy signing oracle return 1; [forgery of a proxy signature by user $\mathtt{1}$ on behalf of user $\mathtt{1}$];
4. if the forgery is of the form $(M, p\Sigma, pk_{\mathtt{1}})$ where $\mathcal{PV}(pk_{\mathtt{1}}, M, p\Sigma) = 1$, and for each message space ω for which $[\mathcal{ID}(p\Sigma), \omega] \in \mathbf{DU}$ it holds that $M \notin \omega$ then return 1; [forgery of a proxy signature by user $i \neq \mathtt{1}$ on behalf of user $\mathtt{1}$; user i was not designated by user $\mathtt{1}$ to sign M];
5. else return 0.

The *advantage* of adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathrm{ps-uf}}_{\mathrm{PS},\mathcal{A}}(\kappa) = \Pr[\mathbf{Exp}^{\mathrm{ps-uf}}_{\mathrm{PS},\mathcal{A}}(\kappa) = 1].$$

We say that PS is a *secure proxy signature scheme* if the function $\mathbf{Adv}^{\mathrm{ps-uf}}_{\mathrm{PS},\mathcal{A}}(\cdot)$ is negligible for all adversaries $\mathcal{A}$ of time complexity polynomial in the security parameter κ.

Appendix B. ECDSA: Attack scenarios

B.1. Signing and verification. We first recall the signing and verification algorithms of ECDSA [23]. Let $(q, FR, a, b, \{, domain\_parameter\_seed\}, G, n, h)$ be the underlying public parameters. Signature generation and verification algorithms are as described below.

signature generation: for message M and key pair (Q, d) where $Q = dG$

1. select random k such that $1 \leq k \leq n-1$
2. compute $kG = [\overline{x}, \overline{y}]$, convert $\overline{x}$ to an integer x
3. compute $r = x \bmod n$, if $r = 0$ go back to Step (1)
4. compute $k^{-1} \bmod n$
5. compute $m = \text{SHA-1}(M)$
6. compute $s = k^{-1}(m + dr) \bmod n$ if $s = 0$ go to Step (1)
7. signature on M is (r, s)

signature verification: of $M, (r, s)$ under public key Q

1. verify (r, s) are integers in $[1, n-1]$.
2. compute $m = \text{SHA-1}(M)$
3. compute $w = s^{-1} \bmod n$
4. compute $u = mw \bmod n$ and $v = rw \bmod n$
5. compute $Z = uG + vQ = s^{-1}mG + s^{-1}rQ$
6. if Z is point at infinity reject, else let $Z = [\tilde{x}_z, \tilde{y}_z]$ and convert $\tilde{x}_z$ to an integer x_z.
7. accept the signature if and only if $r = x_z \bmod n$.

B.2. Toy examples. The toy examples are based on ECDSA as standardized in FIPS186-4 [23]. In the following description

$$(q, FR, a, b, \{, domain\_parameter\_seed\}, G, n, h)$$

will denote the underlying public parameters. We first give an example of Duplicate Signature Key Selection attack (DSKS) following [17, §2.2.4]; then give a demonstration of our attack on delegation-by-certificate construction of proxy signature. The examples also highlight the difference between known DSKS attacks

and our attack on proxy signature. These examples are generated using the SAGE library [29].

B.2.1. *DSKS attack.* For the same message-signature pair, we discuss two variants of the DSKS attack. In DSKS 1 the attacker selects a new generating point but uses the same elliptic curve as the honest signer. In DSKS 2, the attacker selects a new elliptic curve. See Appendix B.3 for details on parameter generation, signing as well as verification of ECDSA. One can also give examples of attacks whereby the underlying finite field is modified but that adds little insight beyond the two attacks and hence is not included.

Suppose the following is a signature generated by an honest user.

Honest signer `1`:

$$\begin{aligned} \text{Parameters} &= (3001, FR, 3, 107, (859 : 628 : 1), 3037, 1), \\ pk_{\mathtt{1}} &= (2757 : 1028 : 1), \\ H(\text{Message}) &= 1308, \\ \text{Signature } (s, r) &= (1701, 524). \end{aligned}$$

DSKS 1: Given the above information, the attacker retains the same elliptic curve but chooses a different generating point. Using $k_2 = 867$, lifting signature value $r = 524$ to a curve point $R = (524 : 1404 : 1)$ and following Steps (5) to (9) in Appendix B.3 results in:

$$\begin{aligned} \text{Parameters} &= (3001, FR, 3, 107, (2043 : 1668 : 1), 3037, 1), \\ pk_{\mathtt{o}} &= (1085 : 2590 : 1), \\ H(\text{Message}) &= 1308, \\ \text{Signature } (s, r) &= (1701, 524). \end{aligned}$$

DSKS 2: In this scenario, the attacker selects a new elliptic curve E_2 of order $\approx q = 3001$, such that E_2 has a point with x coordinate 524. Using $k_2 = 867$, lifting signature value $r = 524$ to a curve point $R = (524 : 111 : 1)$ and following Steps (5) to (9) in Appendix B.3 results in:

$$\begin{aligned} \text{Parameters} &= (3001, FR, 3, 865, (1677 : 226 : 1), 3023, 1), \\ pk'_{\mathtt{o}} &= (2962 : 2420 : 1), \\ H(\text{Message}) &= 1308, \\ \text{Signature } (s, r) &= (1701, 524). \end{aligned}$$

B.2.2. *Proxy signature forgery.* We illustrate the attack on delegation-by-certificate construction of [7] as described in the first attack scenario of §2.

Assume (as in DSKS attacks described above) for user `1` we have:

$$\begin{aligned} \text{Parameters} &= (3001, FR, 3, 107, (859 : 628 : 1), 3037, 1) \\ pk_{\mathtt{1}} &= (2757 : 1028 : 1). \end{aligned}$$

For user `i` we have:

$$\begin{aligned} \text{Parameters} &= (3001, FR, 3, 289, (1874 : 413 : 1), 3049, 1) \\ pk_{\mathtt{i}} &= (1042 : 1548 : 1). \end{aligned}$$

User `i` designates user `1` to proxy sign any message in message space ω. Suppose

$$H(00||1||(2757 : 1028 : 1)||\omega) = 2245$$

then the warrant issued by user $\mathtt{i}$ to user $\mathtt{1}$ will be

$$\mathtt{wrnt}_{\mathtt{i}\to\mathtt{1}} \quad = \quad \mathcal{S}(sk_{\mathtt{i}}, 2245) = (2072, 1048).$$

User $\mathtt{1}$ proxy signs message $M \in \omega$ where we suppose

$$H(01||pk_{\mathtt{i}}||M) \quad = \quad H(01||(1042:1548:1)||M) = 1489.$$

Then proxy signature on message M of user $\mathtt{1}$ on behalf of user $\mathtt{i}$ will be

$$\begin{aligned} p\Sigma &= \mathtt{aux}||\sigma = \mathtt{aux}||\mathcal{S}(sk_{\mathtt{1}}, 1489) = \mathtt{aux}||(2499, 1048) \text{ where} \\ \mathtt{aux} &= \mathtt{1}, \omega, \underbrace{(2757:1028:1)}_{pk_{\mathtt{i}}}, \underbrace{(2072, 1048)}_{\mathtt{wrnt}_{\mathtt{i}\to\mathtt{1}}}. \end{aligned}$$

Using the public information of $\mathtt{i}$, adversary (in the role of user $\mathtt{o}$) picks a random private key

$$sk_{\mathtt{o}} \quad = \quad 2209$$

and computes a new generating point for the same curve as

$$G_o \quad = \quad (sk_{\mathtt{o}})^{-1}\, pk_{\mathtt{i}} = (2755:24:1).$$

User $\mathtt{o}$ sets public information identical to user $\mathtt{i}$ except for the generating point

$$\begin{aligned} \text{Parameters} &= (3001, FR, 3, 289, (2755:24:1), 3049, 1) \\ pk_{\mathtt{o}} &= pk_{\mathtt{i}} = (1042:1548:1). \end{aligned}$$

With the private key $sk_{\mathtt{o}} = 2209$ user $\mathtt{o}$ generates a warrant from user $\mathtt{o}$ to user $\mathtt{i}$ over the same message space ω. From above,

$$H(00||\mathtt{1}||(2757:1028:1)||\omega) \quad = \quad 2245$$

so the warrant issued by user $\mathtt{o}$ to user $\mathtt{1}$ is

$$\mathtt{wrnt}_{\mathtt{o}\to\mathtt{1}} \quad = \quad \mathcal{S}(sk_{\mathtt{o}}, 2245) = (2077, 1471).$$

Then (using publicly available σ from $p\Sigma$) user $\mathtt{o}$ generates

$$\begin{aligned} p\Sigma' &= \mathtt{aux}'||\sigma = \mathtt{aux}'||\mathcal{S}(sk_{\mathtt{1}}, 1489) = \mathtt{aux}'||(2499, 1048) \text{ where} \\ \mathtt{aux}' &= \mathtt{1}, \omega, \underbrace{(2757:1028:1)}_{pk_{\mathtt{1}}}, \underbrace{(2077, 1471)}_{\mathtt{wrnt}_{\mathtt{o}\to\mathtt{1}}} \end{aligned}$$

which is a proxy signature on message M of user $\mathtt{1}$ on behalf of user $\mathtt{o}$.

B.3. KEY SUBSTITUTION ATTACK. Based on ECDSA we show that it is easy for an adversary to create two public keys and two messages that result in the same signature. To do so the adversary creates valid (partial) parameters excluding the generating point G, say

$$(q, FR, a, b, \{, domain\_parameter\_seed\}, \times, n, h).$$

Selects two message M_1 and M_2 ($M_1 = M_2$ is allowed). For $i = 1, 2$

1. compute $m_i = \text{SHA-1}(M_i)$
2. select a random value s such that $1 \le s \le n-1$.
3. select a random curve point $R = [x_r, y_r]$
4. set $r = \text{int}(x_r)$ the integer value of x_r, if $r = 0$ go back to Step (3)
5. select random k_i such that $1 \le k_i \le n-1$
6. compute generating point $G_i = {k_i}^{-1} R$
7. compute private key $d_i = r^{-1}(sk_i - m_i) \bmod n$

8. compute public key $Q_i = d_i G_i$
9. for $i \in \{1, 2\}$ set public parameters and public key to

$$(q, FR, a, b, \{, domain\_parameter\_seed\}, G_i, n, h) \qquad \text{and} \qquad Q_i$$

10. output (s, r) as the signature on message M_i under parameters in the previous step

The signature on the message M_i, $i \in \{1, 2\}$, verifies as

(i) (r, s) are integers in $[1, n-1]$ as selected in Step 2 and Step 4.
(ii) compute $m_i = \text{SHA-1}(M_i)$
(iii) compute $w = s^{-1} \bmod n$
(iv) compute $u = m_i w \bmod n$ and $v = rw \bmod n$
(v) We have that

$$\begin{aligned} Z &= uG_i + vQ_i \\ &= s^{-1}m_iG_i + s^{-1}rQ_i \\ &= s^{-1}m_iG_i + s^{-1}rd_iG_i \qquad \text{by Step 8} \\ &= s^{-1}m_iG_i + s^{-1}rr^{-1}(sk_i - m_i)G_i \qquad \text{by Step 7} \\ &= s^{-1}m_iG_i + s^{-1}(sk_i - m_i)G_i \\ &= s^{-1}m_iG_i + s^{-1}sk_iG_i - s^{-1}m_iG_i \\ &= s^{-1}sk_iG_i \\ &= k_iG_i = k_ik_i^{-1}R = R \qquad \text{by Step 6} \end{aligned}$$

(vi) Since $Z = R$ then it is not the point at infinity and $r = \text{int}(x_z) \bmod n$ validating the signature.

Note that the adversary not only produces the signature, s/he also has the private key corresponding to a public key. The attack can be modified to accommodate different public parameters as long as the x coordinate of the point R is fixed.

APPENDIX C. SECURITY REDUCTIONS

C.1. PROOF OF THEOREM 3.3.

Proof. In the EU-UEO-CMA security game there are more than one public keys from which an adversary may choose a target public key. The argument is a standard reduction with tightness loss of $\frac{1}{n}$ where n is the number of certificates the EU-UEO-UMA adversary is allowed to request.

Given a certificate $\texttt{cert}_{target}$, a corresponding signing oracle and access to an adversary that EU-UEO-CMA breaks the signature scheme we construct the reduction algorithm $\mathcal{R}$. Using $\mathcal{K}$, the algorithm $\mathcal{R}$ generates $n-1$ certificates, and then randomly orders the collection of generated certificates and the input certificate $\texttt{cert}_{target}$. After these preparations $\mathcal{R}$ initiates the EU-UEO-CMA adversary $\mathcal{A}$. When $\mathcal{A}$ requests the kth certificate $\mathcal{R}$ responds with the kth certificate in its list. When $\mathcal{A}$ queries the signing oracle with $\texttt{cert}_{target}$ the algorithm $\mathcal{R}$ queries its own signing oracle with the same message and forwards the answer to $\mathcal{A}$. When $\mathcal{A}$ queries a signing oracle for one of the remaining $n-1$ certificates $\mathcal{R}$ faithfully creates a signature and submits the result to $\mathcal{A}$. When $\mathcal{A}$ requests a private key, the algorithm $\mathcal{R}$ responds faithfully unless the key corresponds to $\texttt{cert}_{target}$ in which case $\mathcal{R}$ aborts with failure.

Suppose $\mathcal{A}$ is successful and completes with output $[(\texttt{cert}_\texttt{i}, M, \sigma), (\texttt{cert}_\texttt{o}, M', \sigma)]$ or output $(\texttt{cert}_\texttt{i}, M, \sigma)$. With probability $\frac{1}{n}$ we have $\texttt{cert}_\texttt{i} = \texttt{cert}_{target}$ in which

case $\mathcal{A}$'s environment is perfectly simulated and $\mathcal{R}$ did not fail. If that is the case, then either $(\mathtt{cert}_\mathtt{i}, M, \sigma)$ is a signature forgery for $\mathtt{cert}_{target}$ breaking the EU-CMA security of the signature scheme or $[(\mathtt{cert}_\mathtt{i}, M, \sigma), (\mathtt{cert}_\mathtt{o}, M', \sigma)]$ breaks the UEO property of the signature scheme. In both cases if $\mathcal{A}$ is successful with probability ϵ the algorithm $\mathcal{R}$ is successful with probability $\frac{\epsilon}{n}$. The running time of $\mathcal{R}$ is the running time of $\mathcal{A}$ plus the time to prepare $(n-1)$ public keys and certificates, respond to signature queries all of which are polynomial in the security parameter. Thus $\mathcal{R}$ runs in polynomial time. Consequently if no polynomial time EU-CMA and no UEO adversary against the signature scheme exists then there is no EU-UEO-CMA adversary against the same signature scheme. □

C.2. SECURITY OF DELEGATION-BY-CERTIFICATE PROXY SIGNATURE.
Conjecture: The proxy signature scheme PS[DS] from §5.1 is a secure proxy signature scheme in the sense of §3.3 if and only if DS is a secure signature scheme in the sense of EU-UEO-CMA security.
Argument: The only if part follows from Theorem 4.2. We will attempt to show that above construction is secure in the sense of §3.3 assuming DS $= (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is a simultaneously UEO- and EU-CMA secure signature scheme. By Theorem 3.3 it suffices to assume that the signature is EU-UEO-CMA secure. The argument requires the construction of a reduction $\mathcal{R}$ that given access to a proxy signature adversary $\mathcal{A}$ breaks the EU-UEO-CMA security of the signature scheme. The algorithm $\mathcal{R}$ initiates $\mathcal{A}$ and responds to $\mathcal{A}$'s queries as follows:

new $\mathtt{i}$**:** $\mathcal{R}$ responds to the query faithfully, except that in the list $\mathbf{PU}_\mathtt{i}$ the entries are of the form $[pk, \mathtt{cert}_\mathtt{o}, \omega, \mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}]$. As $\mathcal{A}$ does not have access to the first entry of $\mathbf{PU}_\mathtt{i}$, the simulation is perfect. Below we explain how proxy signatures are created using the available signature oracles in the simulation;

new $(\mathtt{cert}_\mathtt{i}, \mathrm{info}, \mathtt{i})$**:** $\mathcal{R}$ requests a new public key and crafts a new certificate $\mathtt{cert}_\mathtt{i}$ with the obtained public key, the remaining actions are performed faithfully.

register $\mathtt{cert}_\mathtt{o}$**:** $\mathcal{R}$ responds to the query faithfully;

$\mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}$ **designates** $\mathtt{cert}_\mathtt{o}$**:** $\mathcal{R}$ queries oracle $\mathcal{O}_{\mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}}(\cdot)$ for a signature σ on the message $M = (\mathcal{H}(\mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}) \| \mathcal{H}(\mathtt{cert}_\mathtt{i}^{\mathtt{c_i}} \| \mathtt{cert}_\mathtt{o}) \| \mathcal{H}(\mathtt{cert}_\mathtt{o}) \| \omega)$, sets $\mathtt{wrnt} = \sigma$ and returns the warrant to $\mathcal{A}$. Furthermore, $\mathcal{R}$ sets $\mathbf{DU}_\mathtt{i} = \mathbf{DU}_\mathtt{i} \cup [\mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}, \omega, \mathtt{cert}_\mathtt{o}]$, thus faithfully responding to the query;

$\mathtt{cert}_\mathtt{o}$ **designates** $\mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}$**:** $\mathcal{R}$ responds to the query faithfully except that it appends $[pk_\mathtt{i}^{\mathtt{c_i}}, \mathtt{cert}_\mathtt{o}, \omega, \mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}]$ instead of $[sk_\mathtt{i}^{\mathtt{c_i}}, \mathtt{cert}_\mathtt{o}, \omega, \mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}]$ to $\mathbf{PU}_\mathtt{i}$. As $\mathcal{A}$ does not have access to the first entry of $\mathbf{PU}_\mathtt{i}$ the simulation is perfect;

$\mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}$ **designates** $\mathtt{cert}_\mathtt{j}^{\mathtt{c_j}}$**:** $\mathcal{R}$ responds to the query faithfully except that instead of $[sk_\mathtt{j}^{\mathtt{c_j}}, \mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}, \omega, \mathtt{cert}_\mathtt{j}^{\mathtt{c_j}}]$, an entry $[pk_\mathtt{j}^{\mathtt{c_j}}, \mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}, \omega, \mathtt{cert}_\mathtt{j}^{\mathtt{c_j}}]$ is appended to $\mathbf{PU}_\mathtt{i}$. As $\mathcal{A}$ does not have access to the first entry of $\mathbf{PU}_\mathtt{j}$ the simulation is perfect;

exposure of user $\mathtt{i}$**'s** l**th proxy signing key:** if it exists, $\mathcal{R}$ queries for the signing key corresponding to the public key in the lth entry of $\mathbf{PU}_\mathtt{i}$ and responds to the query faithfully, else the query is ignored;

exposure of certificate $\mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}$ **owned by user** $\mathtt{i}$**:** if it exists, $\mathcal{R}$ queries for the signing key corresponding to the public key in the lth entry of $\mathbf{PU}_\mathtt{i}$ and responds to the query faithfully, else the query is ignored;

standard signature for certificate $\mathtt{cert}_\mathtt{i}^{\mathtt{c_i}}$**:** $\mathcal{R}$ queries the corresponding oracle with the same message and forwards the response to $\mathcal{A}$;

proxy signature of user $\mathtt{i}$ using the lth proxy signing key: given query $(\mathtt{i}, l, M)$, $\mathcal{R}$ verifies that the lth entry of $\mathbf{PU}_{\mathtt{i}}$ is $[pk_{\mathtt{i}}^{\mathtt{c_i}}, \mathtt{cert_o}, \omega, \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}]$ such that $M \in \omega$. If not returns $\perp$, else queries singing oracle $\mathcal{O}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \cdot)$ with the following message: $M' = \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \| \mathtt{cert_o}) \| \mathcal{H}(\mathtt{cert_o}) \| M$ to obtain σ and sets

$$\begin{aligned} p\Sigma &= \sigma = \mathcal{O}_{\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}}(\mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \| \mathtt{cert_o}) \| \mathcal{H}(\mathtt{cert_o}) \| M); \\ \mathtt{aux} &= \mathtt{cert_o} \| \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \| \omega, \mathtt{wrnt} \end{aligned}$$

and forwards the response to $\mathcal{A}$, thus responding faithfully to the query.

Upon completion

1. if $\mathcal{A}$ outputs $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, \sigma)$ where $\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}$ is not compromised, $\mathcal{V}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, \sigma) = 1$, and M was not queried to oracle $\mathcal{O}_{\mathcal{S}}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \cdot)$, then $\mathcal{R}$ successfully outputs $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, \sigma)$ as a signature forgery;
2. if $\mathcal{A}$ outputs $(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}} \to \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, p\Sigma \| \mathtt{aux})$, such that $\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}$ is not compromised, $\mathcal{PV}(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}} \to \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, p\Sigma \| \mathtt{aux}) = 1$, and for all $l \in \mathbb{N}$ with $\mathcal{PU}_{\mathtt{i}}[l] = (spk, \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \omega, \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}})$, no valid $(\mathtt{i}, l, M)$ query was issued to $\mathcal{O}_{\mathcal{PS}}(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}} \to \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \cdot)$ then $\mathcal{R}$ successfully outputs $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M', \sigma)$, where
$$M' = \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \| \mathtt{cert_o}) \| \mathcal{H}(\mathtt{cert_o}) \| M,$$
and $\sigma = p\Sigma$ as a signature forgery;
3. if $\mathcal{A}$ outputs $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \to \mathtt{cert_o}, M, p\Sigma \| \mathtt{aux})$, where
$$\mathtt{aux} = \mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \| \mathtt{cert_o} \| \omega, \mathtt{wrnt}$$
such that for each message space ω for which $[\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \omega, \mathtt{cert_o}] \in \mathbf{DU}_{\mathtt{i}}$ it holds that $M \notin \omega$ then $\mathcal{R}$ returns $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, \mathtt{wrnt})$ where
$$M = \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \| \mathtt{cert_o}) \| \mathcal{H}(\mathtt{cert_o}) \| \omega$$
where $\mathtt{wrnt}$ and $\mathtt{cert_o}$ are the warrant and the proxy certificate under which the proxy signature verifies, thus $\mathcal{R}$ is successful in outputting a signature forgery;
4. if the output is one of the following:
   (a) $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \to \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M, p\sigma \| \mathtt{aux})$ and $(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}} \to \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}, M, p\sigma \| \mathtt{aux}')$ such that at least one of $\{\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}, \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}\}$ is not compromised,
   $$\begin{aligned} \mathcal{PV}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \to \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M, p\sigma \| \mathtt{aux}) &= 1, \\ \mathcal{PV}(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}} \to \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}, M', p\sigma \| \mathtt{aux}') &= 1, \end{aligned}$$
   hold and if in addition one of $\{\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}\}$ is not compromised, then $\mathcal{R}$ returns the UEO forgery $[(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \tilde{M}, p\sigma), (\mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}, \tilde{M}', p\sigma)]$, where
   $$\begin{aligned} \tilde{M} &= \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \| \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}) \| M \\ \tilde{M}' &= \mathcal{H}(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}} \| \mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{j}'}^{\mathtt{c_{j'}}}) \| M'; \end{aligned}$$
   (b) $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \to \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M, p\sigma \| \mathtt{aux})$ and $(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}, M', p\sigma)$ such that at least one certificate from the set $\{\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}\}$ is not compromised,
   $$\begin{aligned} \mathcal{PV}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \to \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M, p\sigma \| \mathtt{aux}) &= 1 \\ \mathcal{V}(\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}, M', p\sigma) &= 1 \end{aligned}$$

hold and if in addition one of $\{\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}\}$ is not compromised then $\mathcal{R}$ returns the UEO forgery $[(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, \tilde{M}, p\sigma), (\mathtt{cert}_{\mathtt{i}'}^{\mathtt{c_{i'}}}, M', p\sigma)]$, where

$$\tilde{M} \quad = \quad \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}} \| \mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}) \| \mathcal{H}(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}) \| M;$$

(c) $(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, p\sigma)$ and $(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M', p\sigma)$ such that at least one certificate is not compromised, $\mathcal{V}(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, p\sigma) = 1$ and $\mathcal{V}(\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M', p\sigma) = 1$, then $\mathcal{R}$ returns the UEO forgery $[(\mathtt{cert}_{\mathtt{i}}^{\mathtt{c_i}}, M, p\sigma), (\mathtt{cert}_{\mathtt{j}}^{\mathtt{c_j}}, M', p\sigma)]$.

REFERENCES

[1] A. Bakker, M. van Steen and A. S. Tanenbaum, A law-abiding peer-to-peer network for free-software distribution, in *IEEE International Symposium on Network Computing and Applications NCA 2001*, Cambridge, MA, USA, IEEE Computer Society, (2001), 60–67.

[2] L. Bassham, W. Polk and R. Housley, Algorithms and identifiers for the internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile, *RFC 3279 (Proposed Standard)*, (2002). Updated by RFCs 4055, 4491, 5480, 5758.

[3] M. Bellare and P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in *11 CCS'93, Proceedings of the 1st ACM Conference on Computer and Communications Security*, Fairfax, Virginia, USA, ACM, (1993), 62–73.

[4] D. J. Bernstein, *Multi-User Schnorr Security, Revisited*, Cryptology ePrint Archive, Report 2015/996, 2015, http://eprint.iacr.org/.

[5] S. Blake-Wilson and A. Menezes, Unknown key-share attacks on the station-to-station (sts) protocol, In *Public Key Cry.Ptography*, (1999), 154–170.

[6] A. Boldyreva, A. Palacio and B. Warinschi, Secure proxy signature schemes for delegation of signing rights, Cryptology ePrint Archive, Report 2003/096, 2003, http://eprint.iacr.org/.

[7] A. Boldyreva, A. Palacio and B. Warinschi, Secure proxy signature schemes for delegation of signing rights, *Journal of Cryptology*, **25** (2012), 57–115.

[8] Certicom Research, *SEC 1: Elliptic Curve Cryptography, Version 2.0*, 2009. Available at: http://www.secg.org/.

[9] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk, Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile, *RFC 5280 (Proposed Standard)*, 2008. Updated by RFC 6818 RFC 8398, RFC 8399.

[10] D. Derler, C. Hanser and D. Slamanig, Privacy-enhancing proxy signatures from non-interactive anonymous credentials, in *Data and Applications Security and Privacy*, **8566** (2014), 49–65.

[11] I. Foster, C. Kesselman, G. Tsudik and S. Tuecke, A security architecture for computational grids, in *CCS '98 Proceedings of the 5th ACM Conference on Computer and Communications Security*, San Francisco, CA, USA, ACM, (1998), 83–92.

[12] S. Galbraith, J. Malone-Lee and N. P. Smart, Public key signatures in the multi-user setting, *Information Processing Letters*, **83** (2002), 263–266.

[13] S. Goldwasser, S. Micali and R. Rivest, A "paradoxical" solution to the signature problem, *Proceedings of the IEEE 25th Annual Symposium on Foundations of Computer Science*, (1984), 441–448.

[14] S. Goldwasser, S. Micali and R. Rivest, A digital signature scheme secure against adaptive chosen-message attacks, *SIAM J. of Computing*, **17** (1988), 281–308.

[15] B. Jens-Matthias, S. Röhrich and R. Steinwandt, Key substitution attacks revisited: Taking into account malicious signers, *International Journal of Information Security*, **5** (2006), 30–36.

[16] E. Kiltz, D. Masny and J. Pan, *Schnorr Signatures in the Multi-User Setting*, Cryptology ePrint Archive, Report 2015/1122, 2015, http://eprint.iacr.org/.

[17] N. Koblitz and A. Menezes, Another look at security definitions, *Advances in Mathematics of Communications*, **7** (2013), 1–38.

[18] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung and C. Wachter, *Ron was Wrong, Whit is Right*, Cryptology ePrint Archive, Report 2012/064, 2012, http://eprint.iacr.org/.

[19] M. Mambo, K. Usuda and E. Okamoto, Proxy signatures for delegating signing operation, in *CCS '96, Proceedings of the 3rd ACM Conference on Computer and Communications Security*, New Delhi, India, ACM, (1996), 48–57.
[20] U. Maurer, Intrinsic limitations of digital signatures and how to cope with them, in *Information Security*, (2003), 180–192.
[21] A. Menezes and N. Smart, Security of signature schemes in a multi-user setting, *Designs, Codes and Cryptography*, **33** (2004), 261–274.
[22] NIST National Institute of Standards and Technology, *Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, 2007. Available via: http://csrc.nist.gov/publications/PubsSPs.html.
[23] NIST National Institute of Standards and Technology, *Digital Signature Standard (DSS) (FIPS 186-4)*, 2013.
[24] T. Pornin and J. P. Stern, Digital signatures do not guarantee exclusive ownership, in *Applied Cryptography and Network Security*, **3531** (2005), 138–150.
[25] M. Stevens, A. Lenstra and B. de Weger, Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities, in *Advances in Cryptology—EUROCRYPT 2007*, Lecture Notes in Comput. Sci., Springer, Berlin, **4515** (2007), 1–22.
[26] M. Stevens, A. Sotirov, J. Appelbaum, A. Lenstra, D. Molnar, D. A. Osvik and B. de Weger, Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate, in *Advances in Cryptology-CRYPTO 2009*, Lecture Notes in Comput. Sci., Springer, Berlin, **5677** (2009), 55–69.
[27] S. Vaudenay, Digital signature schemes with domain parameters: Yet another parameter issue in ECDSA, in *ACISP*, Lecture Notes in Computer Science, Springer, **3108** (2004), 188–199.
[28] P. Yee, Updates to the internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile, *RFC 6818 (Proposed Standard)*, (2013), updates: RFC 5280.
[29] The Sage Developers, *SageMath, the Sage Mathematics Software System (Version 8.0)*, 2017, http://www.sagemath.org.

Received July 2017; revised January 2018.

E-mail address: sanjit@iisc.ac.in
E-mail address: bustaoglu@uwaterloo.ca