The Security Model of Unidirectional Proxy Re-Signature with Private Re-Signature Key

Jun Shao[1,2], Min Feng[3], Bin Zhu[3], Zhenfu Cao[4], and Peng Liu[2]

[1] College of Computer and Information Engineering
Zhejiang Gongshang University
[2] College of Information Sciences and Technology
Pennsylvania State University
[3] Microsoft Research Asia
[4] Department of Computer Science and Engineering
Shanghai Jiao Tong University
chn.junshao@gmail.com, mfeng@microsoft.com, bzhu@microsoft.com,
zfcao@cs.sjtu.edu.cn, pliu@ist.psu.edu

Abstract. In proxy re-signature (PRS), a semi-trusted proxy, with some additional information (a.k.a., re-signature key), can transform Alice's (delegatee) signature into Bob's (delegator) signature on the same message, but cannot produce an arbitrary signature on behalf of either the delegatee or the delegator. In this paper, we investigate the security model of proxy re-signature, and find that the previous security model proposed by Ateniese and Honhenberger at ACM CCS 2005 (referred to as the AH model) is not complete since it does not cover all possible attacks. In particular, the attack on the unidirectional proxy re-signature with private re-signature key. To show this, we artificially design such a proxy re-signature scheme, which is proven secure in the AH model but suffers from a specific attack. Furthermore, we propose a new security model to solve the problem of the AH model. Interestingly, the previous two private re-signature key, unidirectional proxy re-signature schemes (one is proposed by Ateniese and Honhenberger at ACM CCS 2005, and the other is proposed by Libert and Vergnaud at ACM CCS 2008), which are proven secure in the AH model, can still be proven secure in our security model.

Keywords: Security model, Unidirectional PRS, Private re-signature key, AH model.

1 Introduction

1.1 What Is Proxy Re-Signature?

Proxy re-signature (PRS), introduced by Blaze, Bleumer, and Strauss [7], and formalized by Ateniese and Hohenberger [5], allows a semi-trusted proxy to transform a delegatee's (Alice) signature into a delegator's (Bob) signature on the same message by using some additional information (a.k.a., re-signature key).

R. Steinfeld and P. Hawkes (Eds.): ACISP 2010, LNCS 6168, pp. 216–232, 2010.

The proxy, however, cannot generate arbitrary signatures on behalf of either the delegatee or the delegator.

Eight desired properties of proxy re-signatures are given in [5].

1. **Unidirectional:** In a *unidirectional* scheme, a re-signature key allows the proxy to transform A's signature to B's but not vice versa. In a *bidirectional* scheme, on the other hand, the re-signature key allows the proxy to transform A's signature to B's as well as B's signature to A's.
2. **Multi-use:** In a *multi-use* scheme, a transformed signature can be re-transformed again by the proxy. In a *single-use* scheme, the proxy can transform only the signatures that have not been transformed.
3. **Private re-signature key:** The proxy can keep the re-signature key as a secret in a *private re-signature key* scheme, but *anyone* can recompute the re-signature key by observing the proxy passively in a *public re-signature key* scheme.
4. **Transparent:** In a *transparent* scheme, users may not even know that a proxy exists.
5. **Key-optimal:** In a *key-optimal* scheme, a user is required to protect and store only a small constant amount of secrets no matter how many signature delegations the user gives or accepts.
6. **Non-interactive:** The delegatee is not required to participate in delegation process.
7. **Non-transitive:** A re-signing right cannot be re-delegated by the proxy alone.
8. **Temporary:** A re-signing right is temporary. This can be done by either revoking the right as in [5] or expiring the right.

1.2 Applications of PRS

One interesting application of PRS is the interoperable architecture of ditigal rights management (DRM). A DRM system is designed to prevent illegal redistribution of digital content. With DRM systems, the digital content can only be played in a specified device (regime). For example, a song playable in device (regime) A cannot be played in device (regime) B. However, it is reported that 86% consumers prefer paying twice price for a song that runs on any device rather than runs on only one device [2]. Most of current interoperability architectures require to change the existing DRM systems a lot [11]; this modification cannot be adopted due to business reasons. Based on proxy re-signature and proxy re-encryption [3,4], Taban *et al.* [14] proposed a new interoperability architecture, which does not change the existing DRM systems a lot but keep the DRM systems' security. In their architecture, only a new module called Domain Interoperability Manager (DIM) is introduced. PRS allows the DIM to transform licenses (signatures) in regime A into another in regime B, while DIM cannot generate valid licenses (signatures) either in regime A or in regime B.

PRS can also be used in other applications according to its properties: Space-efficient proof that a path was taken (by using the multi-usability and private

re-signature key properties) [5], management of group signatures (by using the unidirection and private re-signature key properties) [5]. We refer the reader to [1] for more applications.

1.3 History of Proxy Re-Signatures

Proxy re-signature has a rather short history. This primitive was introduced at Eurocrypt 1998 by Blaze, Bleumer and Strauss [7]. Their scheme, referred to as the BBS scheme, is a *multi-use*, *public re-signature key* and *bidirectional* scheme. However, from the re-signature key (which is public), the delegator can easily obtain the delegatee's signing key or vice versa. As a result, the BBS scheme is not suitable for many practical applications described in [7].

After the first proxy re-signature scheme appeared in 1998, there was no follow-ups until the work by Ateniese and Hohenberger published at ACM CCS 2005 [5]. One of the reasons for such a long quiet time is, as stated in [5], that the definition of proxy re-signatures given in [7] is informal and can be easily confused with other signature variations. In [5], Ateniese and Hohenberger first formalized the definition of security for a proxy re-signature, referred to as the AH model in this paper, and then proposed three proxy re-signature schemes with proven security. The first one is *multi-use*, *private re-signature key* and *bidirectional*, the second one is *single-use*, *public re-signature key* and *unidirectional*, and the third one is *single-use*, *private re-signature key* and *unidirectional*. Later, Shao *et al.* [13] proposed a *multi-use*, *private re-signature key* and *bidirectional* proxy re-signature scheme based on Waters' identity based signature [15], and Libert and Vergnaud [12] proposed a *multi-use*, *private re-signature key* and *unidirectional* scheme based on the ℓ-FlexDH assumption. All the proxy re-signature schemes in [5,13,12] are proven secure in the AH model.

1.4 This Paper's Contributions

The AH model covers two types of forgeries for bidirectional proxy re-signature: (1) an outsider who is neither the proxy nor one of the delegation parties aims to produce signatures on behalf of either delegation party; (2) the proxy aims to produce signatures on behalf of either delegation party. For unidirectional proxy re-signature, in addition to these two types of forgeries, the AH model covers another two types of forgeries: (3) the delegator colludes with the proxy to produce signatures on behalf of the delegatee; (4) the delegatee colludes with the proxy to produce the *first-level* signatures[1].

The AH model covers almost all types of forgeries for unidirectional proxy re-signatures, but it omits one type of forgeries that the delegatee may aim to produce signatures on behalf of the delegator *without* colluding with the proxy. For example, for the transformation path: Alice $\rightarrow$ Proxy $\rightarrow$ Bob. Alice may attempt to produce a *second-level* signature on a message on behalf of Bob without the transformation of Proxy. This situation is not allowed in the proxy

[1] See Remark 1 in Section 2.

re-signature schemes with private re-signature key, since Bob has delegated his signing rights via Proxy but not to Alice directly. The schemes suffering from this attack cannot be used in most of the applications of PRS listed in [1].

To show the deficiency of the AH model, we artificially design in Section 3.3 a scheme, denoted as S_{us}, which suffers from the above attack but is proven secure in the AH model.

On one hand, one of the possible reasons for the deficiency of the AH model is that the AH model tried to model *all* types of attacks on *all* types of proxy re-signatures. Hence, it is more complex than security models of other types of signatures, which makes the AH model hard to be verified. It would be better if one security model is associated to only one type of proxy re-signatures.

On the other hand, as shown in [1], the unidirectional proxy re-signature with private re-signature key is more useful than other types of proxy re-signatures. Hence, it is desired to propose a security model to instruct people to design the unidirectional proxy re-signature with private re-signature key.

As a result, in this paper, we only focus on the security model of the unidirectional proxy re-signature with private re-signature key. The proposed security model is clearer and simpler than the AH model (one security game vs. four security games). For the simple expression, we refer to this type of proxy re-signatures as `UPRS-prk` in the rest of this paper. Fortunately, the previous `UPRS-prk`[2] schemes [5,12], which are proven secure in the AH model, can still be proven secure in our model.

1.5 Paper Organization

The remaining paper is organized as follows. In Section 2, the definitions of `UPRS-prk` are introduced and the AH model is described. In Section 3, we present a scheme which is proven secure in the AH model but insecure. In Section 4, we present our security model of `UPRS-prk` and point out that the previous `UPRS-prk` schemes are still secure in our model. We conclude the paper in Section 5.

2 Definitions

2.1 Unidirectional Proxy Re-Signature with Private Re-Signature Key

The following definitions are from [5].

Definition 1 (`UPRS-prk`)**.** *A `UPRS-prk` scheme `PRS` consists of the following five probabilistic algorithms:* `KeyGen`*,* `ReKey`*,* `Sign`*,* `ReSign`*, and* `Verify` *where:*

`KeyGen`**:** *It takes as input the security parameter* 1^k*, and returns a verification key* pk *and a signing key* sk*. This algorithm is denoted as* $(pk, sk) \leftarrow$ `KeyGen`(1^k).

[2] All existing `UPRS-prk` schemes are non-interactive. Hence, we only focus on non-interactive `UPRS-prk` in this paper.

`ReKey`: *It takes as input delegatee Alice's verification key* pk_A*, and delegator Bob's key pair* (pk_B, sk_B)*, and returns a re-signature key* $rk_{A\to B}$ *for the proxy. This algorithm is denoted as* $rk_{A\to B} \leftarrow \texttt{ReKey}(pk_A, pk_B, sk_B)$.

`Sign`: *It takes as input a signing key* sk*, a positive integer* ℓ *and a message* m*, and returns a signature* σ *at level* ℓ*. This algorithm is denoted as* $\sigma \leftarrow \texttt{Sign}(sk, m, \ell)$.

`ReSign`: *It takes as input a re-signature key* $rk_{A\to B}$*, a signature* σ_A *on a message* m *under* pk_A *at level* ℓ*, and returns the signature* σ_B *on the same message* m *under* pk_B *at level* $\ell + 1$ *if* $\texttt{Verify}(pk_A, m, \sigma_A, \ell) = 1$*, or* $\perp$ *otherwise. This algorithm is denoted as* $\sigma_B \leftarrow \texttt{ReSign}(rk_{A\to B}, pk_A, m, \sigma_A, \ell)$.

`Verify`: *It takes as input a verification key* pk*, a message* m*, a signature* σ *and a positive integer* ℓ*, and returns* 1 *if* σ *is a valid signature under* pk *at level* ℓ*, or 0 otherwise. This algorithm is denoted as* $(1 \text{ or } 0) \leftarrow \texttt{Verify}(pk, m, \sigma, \ell)$.

Correctness. The following property must be satisfied for the correctness of a `UPRS-prk` scheme: For any message m in the message space and any two key pairs (pk_A, sk_A) and (pk_B, sk_B), let $rk_{A\to B} \leftarrow \texttt{Rekey}(pk_A, pk_B, sk_B)$, the following two equations must hold:

$$\texttt{Verify}(pk_A, m, \sigma_A, \ell) = 1,$$

where σ_A is a signature on message m under pk_A at level ℓ from `Sign`. If the `UPRS-prk` scheme is single-use, then $\ell \in \{1, 2\}$; $\ell \geq 1$ otherwise.

$$\texttt{Verify}(pk_B, m, \texttt{ReSign}(rk_{A\to B}, pk_A, m, \sigma'_A, \ell - 1), \ell) = 1.$$

If the `UPRS-prk` scheme is single-use, σ'_A is a signature on message m under pk_A from `Sign`, and $\ell = 2$; if the proxy re-signature scheme is multi-use, σ'_A could also be a signature on message m under pk_A from `ReSign`, and $\ell \geq 2$.

Remark 1 (Two Types of Signatures.). In all existing unidirectional proxy re-signature schemes, a signature manifests in two types: the *owner-type* (i.e., the first-level defined in [5], $\ell = 1$) and the *non-owner-type* (i.e., the second-level signatures in [5], $\ell > 1$). An *owner-type* signature can be computed only by the owner of the signing key, while a *non-owner-type* signature can be computed not only by the owner of the signing key, but also by collaboration between his proxy and delegatee.

2.2 The AH Model

In this subsection, we review the AH model for `UPRS-prk`. It contains two aspects (four security games): the external security and the internal security. The details are as follows.

External Security: This security deals with adversaries other than the proxy and any delegation parties. A `UPRS-prk` scheme has external security if and

only if for security parameter k, any non-zero $n \in \text{poly}(k)$, and all probabilistic polynomial time (p.p.t.) algorithms $\mathcal{A}$, the following probability is *negligible*:

$$\Pr[\{(pk_i, sk_i) \leftarrow \texttt{KeyGen}(1^k)\}_{i\in[1,n]}, (t, m^*, \sigma^*, \ell^*) \leftarrow \mathcal{A}^{\mathcal{O}_s(\cdot),\mathcal{O}_{rs}(\cdot)}(\{pk_i\}_{i\in[1,n]}) : \texttt{Verify}(pk_t, m^*, \sigma^*, \ell^*) = 1 \wedge (t, m^*) \notin Q],$$

where oracle $\mathcal{O}_s$ takes as input a verification key pk_i and a message $m \in \mathcal{M}$, and produces an output $\texttt{Sign}(sk_i, m, 1)$; oracle $\mathcal{O}_{rs}$ takes as input two distinct verification keys pk_i and pk_j, a message m, a signature σ and a positive integer ℓ, and produces an output $\texttt{ReSign}(\texttt{ReKey}(pk_i, pk_j, sk_j), pk_i, m, \sigma, \ell)$; and Q denotes the set of $(index, message)$ pairs (i, m) that $\mathcal{A}$ obtains a signature on a message m under the verification key pk_i by querying $\mathcal{O}_s$ on (pk_i, m) or querying $\mathcal{O}_{rs}$ on $(\cdot, pk_i, m, \cdot, \cdot)$. Note that if the treated PRS scheme is single-use, then $\ell^* \in \{1, 2\}$ and $\ell = 1$; otherwise, $\ell^* \geq 1$ and $\ell \geq 1$.

Internal Security: This security protects a user from inside adversaries who can be any parties, i.e., the proxy, the delegatee, and the delegator, in a proxy re-signature scheme. It can be classified into the following three types.

`Limited Proxy:` In this case that only the proxy is a potential adversary $\mathcal{A}$. We must guarantee that the proxy cannot produce signatures on behalf of either the delegator or the delegatee except the signatures produced by the delegatee and delegated to the proxy to re-sign. Internal security in this case is very similar to external security described above except that $\mathcal{A}$ queries a rekey oracle $\mathcal{O}_{rk}$ instead of a re-signature oracle $\mathcal{O}_{rs}$. A `UPRS-prk` scheme is said to have limited proxy security if and only if for security parameter k, any non-zero $n \in \text{poly}(k)$, and all p.p.t. algorithms $\mathcal{A}$, the following probability is *negligible*:

$$\Pr[\{(pk_i, sk_i) \leftarrow \texttt{KeyGen}(1^k)\}_{i\in[1,n]}, (t, m^*, \sigma^*, \ell^*) \leftarrow \mathcal{A}^{\mathcal{O}_s(\cdot),\mathcal{O}_{rk}(\cdot)}(\{pk_i\}_{i\in[1,n]}) : \texttt{Verify}(pk_t, m^*, \sigma^*, \ell^*) = 1 \wedge (t, m^*) \notin Q],$$

where $\mathcal{O}_s$ and ℓ^* are the same as that in **external security**, oracle $\mathcal{O}_{rk}$ takes as input two distinct verification keys pk_i, pk_j, and returns the output of $\texttt{ReKey}(pk_i, pk_j, sk_j)$; and Q denotes the set of $(index, message)$ tuples (i, m) that $\mathcal{A}$ obtained a signature on m under verification key pk_i or one of its delegatees' keys by querying $\mathcal{O}_s$.

`Delegatee Security:` In this case that the proxy and delegator may collude with each other. This security guarantees that their collusion cannot produce any signatures on behalf of the delegatee. We associate the index 0 to the delegatee. A `UPRS-prk` scheme is said to have delegatee security if and only if for security parameter k, any non-zero $n \in \text{poly}(k)$, and all p.p.t. algorithms $\mathcal{A}$, the following probability is *negligible*:

$$\Pr[\{(pk_i, sk_i) \leftarrow \texttt{KeyGen}(1^k)\}_{i\in[0,n]}, (m^*, \sigma^*, \ell^*) \leftarrow \mathcal{A}^{\mathcal{O}_s(\cdot)}(pk_0, \{pk_i, sk_i\}_{i\in[1,n]}) : \texttt{Verify}(pk_0, m^*, \sigma^*, \ell^*) = 1 \wedge (0, m^*) \notin Q],$$

where $\mathcal{O}_s$ and ℓ^* are the same as that in `external security`, Q is the set of pairs $(0, m)$ that $\mathcal{A}$ obtains a signature by querying oracle $\mathcal{O}_s$ on (pk_0, m). Note that $\mathcal{O}_{rk}$ is useless in this case, since the adversary is able to compute re-signature keys $rk_{0\to i}$ himself by using sk_i.

`Delegator Security:` In this case that the proxy and delegatee may collude with each other. This security guarantees that their collusion cannot produce any owner-type signatures on behalf of the delegator. We associate the index 0 to the delegator. A `UPRS-prk` scheme is said to have delegator security if and only if for security parameter k, any non-zero $n \in \text{poly}(k)$, and all p.p.t. algorithms $\mathcal{A}$, the following probability is *negligible*:

$$\Pr[\{(pk_i, sk_i) \leftarrow \texttt{KeyGen}(1^k)\}_{i\in[0,n]}, (m^*, \sigma^*, 1) \leftarrow \mathcal{A}^{\mathcal{O}_s(\cdot),\mathcal{O}_{rk}(\cdot)}(pk_0, \{pk_i, sk_i\}_{i\in[1,n]}) : \texttt{Verify}(pk_0, m^*, \sigma^*, 1) = 1 \wedge (0, m^*) \notin Q],$$

where $\mathcal{O}_s$ is the same as that in **external security**, $\mathcal{O}_{rk}$ is the same as that in `limited proxy security`, Q is the set of pairs $(0, m)$ for which $\mathcal{A}$ obtains an owner-type signature by querying oracle $\mathcal{O}_s$ on (pk_0, m).

Remark 2. According the definition in [5] (page 313), we say a unidirectional proxy re-signature scheme is *public re-signature key* if it does not hold the external security, like scheme S_{uni} in [5]; otherwise, the scheme is private re-signature key, like scheme S^*_{uni} in [5] and the schemes in [12].

3 On the AH Model

Before proposing scheme S_{us}, we first introduce some basic knowledge related to scheme S_{us}.

3.1 Bilinear Groups

Bilinear maps and bilinear map groups are briefly reviewed in this subsection. Details can be found in [8,9].

1. $\mathbb{G}$ and $\mathbb{G}_T$ are two (multiplicative) cyclic groups of prime order q;
2. g is a generator of $\mathbb{G}$;
3. e is a bilinear map, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups as above. An *admissible bilinear map* is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

1. *Bilinearity*: For all $P, Q, R \in \mathbb{G}$, $e(P \cdot Q, R) = e(P, R) \cdot e(Q, R)$ and $e(P, Q \cdot R) = e(P, Q) \cdot e(P, R)$.
2. *Non-degeneracy*: If $e(P, Q) = 1$ for all $Q \in \mathbb{G}$, then $P = \mathcal{O}$, where $\mathcal{O}$ is a point at infinity.

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exist a group $\mathbb{G}_T$ and an efficiently computable bilinear map as above. We use `BSetup` to denote an algorithm that, on input the security parameter 1^k, outputs the parameters for a bilinear map as $(q, g, \mathbb{G}, \mathbb{G}_T, e)$, where $q \in \Theta(2^k)$.

3.2 Complexity Assumption

The security of scheme S_{us} can be proved based on the extended Computational Diffie-Hellman (eCDH) assumption in the AH model.

Definition 2 (Extended Computational Diffie-Hellman Assumption). *Let* $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathtt{BSetup}(1^k)$. *The extended computational Diffie-Hellman problem (eCDH) in* $(\mathbb{G}, \mathbb{G}_T)$ *is defined as follows: given 4-tuple* $(g, g^u, g^v, g^{1/v}) \in \mathbb{G}^5$ *as input, output* g^{uv} *or* $g^{u/v}$. *An algorithm* $\mathcal{A}$ *has advantage* ε *in solving the eCDH problem in* $(\mathbb{G}, \mathbb{G}_T)$ *if*

$$Pr[\mathcal{A}(g, g^u, g^v, g^{1/v}) = g^{uv} \text{ or } g^{u/v}] \geq \varepsilon,$$

where the probability is taken over the random choices of $u, v \in Z_q^*$ *and the random bits of* $\mathcal{A}$.

We say that the (t, ε)*-extended computational Diffie-Hellman (eCDH) assumption holds in* $(\mathbb{G}, \mathbb{G}_T)$ *if no* t*-time algorithm has advantage* ε *at least in solving the eCDH problem in* $(\mathbb{G}, \mathbb{G}_T)$.

In this paper, we drop the t and ε and refer to the eCDH assumption rather than the (ε, t)-eCDH assumption.

3.3 The Scheme S_{us}

In this subsection, we propose a `UPRS-prk` scheme, named S_{us}, which is proven secure in the AH model, but it cannot provide all the required security properties. This fact shows that the AH model is not complete. The public parameters of scheme S_{us} are $(q, g, \mathbb{G}, \mathbb{G}_T, e)$, where $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathtt{BSetup}(1^k)$, H is a cryptographic hash function: $\{0,1\}^* \rightarrow \mathbb{G}$.

`KeyGen:` On input the security parameter 1^k, it selects a random number $a \in Z_q^*$, and outputs the key pair $pk = g^a$ and $sk = a$.

`ReKey:` On input the delegatee's verification key $pk_A = g^a$ and the delegator's signing key $sk_B = b$, it outputs the re-signature key

$$rk_{A\rightarrow B} = (rk^{(1)}_{A\rightarrow B}, rk^{(2)}_{A\rightarrow B}, rk^{(3)}_{A\rightarrow B}) = (r', (pk_A)^{r'}, H(g^{a\cdot r'}||2)^{1/b}),$$

where r' is a random number in Z_q^* determined by Bob.

`Sign:` On input a signing key $sk = a$, a message $m \in \mathcal{M}$ and an integer $\ell \in \{1, 2\}$,

- if $\ell = 1$, it outputs an *owner-type* signature

$$\sigma = (A, B, C) = (H(m||0)^r, g^r, H(g^r||1)^a),$$

- if $\ell = 2$, it outputs a *non-owner-type* signature

$$\sigma = (A, B, C, D, E) = (H(m||0)^{r_1}, g^{r_1}, H(g^{r_1}||1)^{r_2}, g^{r_2}, H(g^{r_2}||2)^{1/a}).$$

ReSign: Given an *owner-type* signature σ at level 1, a re-signature key $rk_{A\to B} = (rk^{(1)}_{A\to B}, rk^{(2)}_{A\to B}, rk^{(3)}_{A\to B})$, a verification key pk_A, and a message m, this algorithm first checks $\texttt{Verify}(pk_A, m, \sigma, 1) \stackrel{?}{=} 1$. If it does not hold, outputs $\perp$; otherwise, outputs

$$\begin{array}{lllll}
\sigma' = (A', & B', & C', & D', & E') \\
= (A, & B, & C^{rk^{(1)}_{A\to B}}, & rk^{(2)}_{A\to B}, & rk^{(3)}_{A\to B}) \\
= (H(m||0)^{r}, & g^{r}, & H(g^{r}||1)^{ar'}, & (pk_A)^{r'}, & H((pk_A)^{r'}||2)^{1/b}) \\
= (H(m||0)^{r_1}, & g^{r_1}, & H(g^{r_1}||1)^{r_2}, & g^{r_2}, & H(g^{r_2}||2)^{1/b})
\end{array}$$

Note that we set $r_1 = r \bmod q$ and $r_2 = ar' \bmod q$.

Verify: On input a verification key pk, a message m at level $\ell \in \{1, 2\}$, and a signature σ,

- if σ is an *owner-type* signature $\sigma = (A, B, C)$ (i.e., $\ell = 1$), it checks

$$e(pk, H(B||1)) \stackrel{?}{=} e(g, C),$$
$$e(B, H(m||0)) \stackrel{?}{=} e(g, A).$$

  If the two equations both hold, it outputs 1; otherwise, outputs 0.
- if σ is a *non-owner-type* signature $\sigma = (A, B, C, D, E)$ (i.e., $\ell = 2$), it checks

$$e(g, H(D||2)) \stackrel{?}{=} e(pk, E),$$
$$e(D, H(B||1)) \stackrel{?}{=} e(g, C),$$
$$e(B, H(m||0)) \stackrel{?}{=} e(g, A).$$

  If all the equations hold, it outputs 1; otherwise, outputs 0.

3.4 Correctness

Scheme S_{us} has the correctness due to the following equations.

- owner-type signature:

$$e(pk, H(B||1)) = e(g^a, H(B||1)) = e(g, H(B||1)^a) = e(g, C),$$
$$e(B, H(m||0)) = e(g^r, H(m||0)) = e(g, H(m||0)^r) = e(g, A),$$

- non-owner-type signatures:

$$e(pk, E) = e(g^b, H(D||2)^{1/b}) = e(g, H(D||2)),$$
$$e(D, H(B||1)) = e(g^{r_2}, H(B||1)) = e(g, H(B||1)^{r_2}) = e(g, C),$$
$$e(B, H(m||0)) = e(g^{r_1}, H(m||0)) = e(g, H(m||0)^{r_1}) = e(g, A).$$

3.5 Security Analysis

Theorem 1. *Scheme S_{us} is secure in the AH model if the eCDH problem is hard, and hash function H is treated as a random oracle.*

Proof. We prove the security in the AH model in two parts, similar to that in [5].

We show that if adversary $\mathcal{A}$ can break scheme S_{us} in the AH model, then we can build another algorithm $\mathcal{B}$ that can solve the eCDH problem. Given $(q, g, \mathbb{G}, \mathbb{G}_T, e, g^u, g^v, g^{1/v})$, $\mathcal{B}$ aims to output g^{uv} or $g^{u/v}$. The proxy re-signature security game is as follows.

External Security.

- *Random oracle $\mathcal{O}_h$:* On input string R, $\mathcal{B}$ first checks whether $(R, R_h, r_h, *)$ exists in Table T_h. If yes, $\mathcal{B}$ returns R_h and terminates; otherwise, $\mathcal{B}$ chooses a random number $r_h \in Z_q^*$, and the next performance of $\mathcal{B}$ has three situations.
  - The input string R satisfies the format $m||0$, where $m \in \mathcal{M}$. $\mathcal{B}$ guesses whether m is the target message m^*. If yes, $\mathcal{B}$ outputs $R_h = (g^u)^{r_h}$; otherwise, $\mathcal{B}$ outputs $R_h = g^{r_h}$.
  - The input string R satisfies the format $m||1$ or $m||2$, where $m \in \mathbb{G}$. $\mathcal{B}$ outputs $R_h = (g^u)^{r_h}$.
  - The input string R does not satisfy any of the above formats. $\mathcal{B}$ outputs $R_h = g^{r_h}$.

  At last, $\mathcal{B}$ records $(R, R_h, r_h, \perp)$ in Table T_h.
- *Verification keys oracle $\mathcal{O}_{pk}$:* As the adversary requests the creation of system users, $\mathcal{B}$ first chooses a random number $x_i \in Z_q^*$, and guesses whether it is pk_t. For $pk_i \neq pk_t$, it sets $pk_i = g^{x_i}$; for $pk_i = pk_t$, it sets $pk_i = (g^v)^{x_i}$. At last, $\mathcal{B}$ records (pk_i, x_i) in Table T_{pk}.
- *Signature oracle $\mathcal{O}_s$:* On input (pk_i, m_i).
  - If $m_i = m^*$, then $pk_i \neq pk_t$, $\mathcal{B}$ chooses a random number $r \in Z_q^*$, and outputs

$$\sigma = (A, B, C) = (H(m^*||0)^r, g^r, H(g^r||1)^{x_i}).$$

  - If $m_i \neq m^*$, then $\mathcal{B}$ chooses a random number $r \in Z_q^*$, and checks whether $((g^v)^r||1, *, *, *)$ exists in Table T_h. If it exists, $\mathcal{B}$ reports "failure" and aborts; otherwise $\mathcal{B}$ chooses a random number $r_1 \in Z_q^*$, and records $((g^v)^r||1, g^{r_1}, r_1, r)$ in Table T_h. And then $\mathcal{B}$ checks whether $(m_i||0, \star_1, \star_2, \perp)$ exists in Table T_h. If it exists, then $\mathcal{B}$ sets $r_2 = \star_2$; otherwise, $\mathcal{B}$ chooses a random number $r_2 \in Z_q^*$ and records $(m_i||0, g^{r_2}, r_2, \perp)$ in Table T_h. At last, $\mathcal{B}$ outputs

$$\sigma = (A, B, C) = ((g^v)^{rr_2}, (g^v)^r, pk_i^{r_1}) = (H(m_i||0)^{vr}, g^{vr}, H(g^{vr}||1)^{vx_i}).$$

- *Re-signature oracle $\mathcal{O}_{rs}$:* On input $(pk_i, pk_j, m, \sigma, 1)$, where $\sigma = (A, B, C)$. If $\texttt{Verify}(pk_i, m, \sigma, 1) = 1$, then B does the following performances; otherwise, outputs $\perp$.

- If $pk_j \neq pk_t$, $\mathcal{B}$ uses x_j, associated to pk_j in Table T_{pk}, to run `ReKey` and `ReSign`, and gets the required re-signature.
- If $pk_j = pk_t$, then $m \neq m^*$, and $\mathcal{B}$ chooses a random number $r \in Z_q^*$, and checks whether $((g^v)^{x_i r}||2, *, *, *)$ exists in Table T_h. If it exists, $\mathcal{B}$ reports "failure" and aborts; $\mathcal{B}$ chooses a random number $r_1 \in Z_q^*$, and records $((g^v)^{x_i r}||2, g^{r_1}, r_1, x_i r)$ in Table T_h. $\mathcal{B}$ searches $(B||1, \star_1, \star_2, \star_3)$ in Table T_h, and outputs

$$\begin{aligned}\sigma' &= (\ A',\ B',\ C', && D', && E')\\ &= (\ A,\ B,\ (g^v)^{x_i \star_2 r}, && (g^v)^{x_i r}, && (g^{1/v})^{r_1/x_t})\\ &= (\ A,\ B,\ H(B||1)^{x_i v r}, && g^{x_i v r}, && H(g^{x_i v r}||2)^{1/(v x_t)}),\end{aligned}$$

– *Forgery:* At some point, the adversary must output a forgery (pk_t, m^*, σ^*).

Now, we show how $\mathcal{B}$ gets the eCDH solution from the forgery.

– If σ^* is an owner-type signature, such as $\sigma^* = (A^*, B^*, C^*)$, then we have the following analysis.
  - If $(*, B^*, C^*)$ did not appear in one owner-type signature of pk_t from $\mathcal{O}_s$, then $\mathcal{B}$ finds $(B^*||1, \star_1', \star_2', \star_3')$ in Table T_h, and gets the solution of the eCDH problem:

$$(C^*)^{1/(x_t \star_2')} = (H(B^*||1)^{v x_t})^{1/(x_t \star_2')} = (g^{u \star_2' v x_t})^{1/(x_t \star_2')} = g^{uv}.$$

    Note that $pk_t = (g^v)^{x_t}$.
  - If $(*, B^*, C^*)$ appeared in one owner-type signature of pk_t from $\mathcal{O}_s$, then $\mathcal{B}$ finds $(m^*||0, \star_1, \star_2, \star_3)$ and $(B^*||1, \star_1', \star_2', \star_3')$ in Table T_h, and gets the solution of the eCDH problem:

$$(A^*)^{1/(\star_2 \star_3')} = (H(m^*||0)^{v \star_3'})^{1/(\star_2 \star_3')} = (g^{uv \star_2 \star_3'})^{1/(\star_2 \star_3')} = g^{uv}.$$

    Note that $B^* = (g^v)^{\star_3'}$.
– If σ^* is a non-owner-type signature, such as $\sigma^* = (A^*, B^*, C^*, D^*, E^*)$, then we have the following analysis.
  - If $(*, *, *, D^*, E^*)$ did not appear in any one signature of pk_t from $\mathcal{O}_{rs}$, then $\mathcal{B}$ finds $(D^*||2, \star_1'', \star_2'', \star_3'')$ in Table T_h, and gets the solution of the eCDH problem:

$$(E^*)^{x_t/\star_2''} = (H(D^*||2)^{1/(v x_t)})^{x_t/\star_2''} = (g^{u \star_2''/(v x_t)})^{x_t/\star_2''} = g^{u/v}.$$

    Note that $pk_t = (g^v)^{x_t}$.
  - If $(*, *, *, D^*, E^*)$ appeared in one signature of pk_t, but $(*, B^*, C^*, D^*, E^*)$ did not appear in any one signature of pk_t from $\mathcal{O}_{rs}$, then $\mathcal{B}$ finds $(D^*||2, \star_1'', \star_2'', \star_3'')$ and $(B^*||1, \star_1', \star_2', \star_3')$ in Table T_h, and gets the solution of the eCDH problem:

$$(C^*)^{1/(\star_3'' \star_2')} = (H(B^*||1)^{v \star_3''})^{1/(\star_3'' \star_2')} = (g^{u \star_2' v \star_3''})^{1/(\star_3'' \star_2')} = g^{uv}.$$

    Note that $D^* = (g^v)^{\star_3''}$.

- If $(*, B^*, C^*, D^*, E^*)$ appeared in one owner-type signature of pk_t from $\mathcal{O}_{rs}$, then $\mathcal{B}$ finds $(m^*||0, \star_1, \star_2, \star_3)$ and $(B||1, \star_1', \star_2', \star_3')$ in Table T_h, and gets the solution of the eCDH problem:

$$(A^*)^{1/(\star_2\star_3')} = (H(m^*||0)^{v\star_3'})^{1/(\star_2\star_3')} = (g^{uv\star_2\star_3'})^{1/(\star_2\star_3')} = g^{uv}.$$

  Note that $B^* = (g^v)^{\star_3'}$.

Note that $\mathcal{B}$ guessed the right target verification key with the probability $1/n$ at least, and $\mathcal{B}$ reports "failure" and aborts in $\mathcal{O}_s$ and $\mathcal{O}_{rs}$ with the probabilities $(q_h + q_s)/q$ and $(q_h + q_{rs})/q$ at most, respectively. Here, q_h, q_s, and q_{rs} are the maximum numbers that $\mathcal{A}$ can query to random oracle $\mathcal{O}_h$, signature oracle $\mathcal{O}_s$, re-signature oracle $\mathcal{O}_{rs}$ respectively. As a result, $\mathcal{B}$ solves the eCDH problem with a non-negligible probability.

Internal Security: Internal security includes three parts: Limited Proxy Security, Delegatee Security, Delegator Security.

Limited Proxy Security:

- *Random oracle $\mathcal{O}_h$:* Identical to that in the external security.
- *Verification keys oracle $\mathcal{O}_{pk}$:* As the adversary requests the creation of system users, $\mathcal{B}$ first chooses a random number $x_i \in Z_q^*$, and then outputs $pk_i = (g^v)^{x_i}$. At last, $\mathcal{B}$ records (pk_i, x_i) in Table T_{pk}.
- *Signature oracle $\mathcal{O}_s$:* On input (pk_i, m_i).
  - If $m = m^*$, then $pk_i \neq pk_t$, and $\mathcal{B}$ chooses a random number r, and checks whether $(g^r||1, *, *, *)$ exists in Table T_h. If it exists, then $\mathcal{B}$ reports "failure" and aborts; otherwise, $\mathcal{B}$ chooses a random number r_1, and records $(g^r||1, g^{r_1}, r_1, \perp)$ into Table T_h. At last, $\mathcal{B}$ outputs $(H(m^*||0)^r, g^r, pk_i^{r_1})$.
  - If $m \neq m^*$, then $\mathcal{B}$ performs the same as that in the external security.
- *Re-signature key generation oracle $\mathcal{O}_{rk}$:* On input (pk_i, pk_j), $\mathcal{B}$ chooses a random number $r \in Z_q^*$, and checks whether $((g^v)^{x_i r}||2, *, *, *)$ exists in Table T_h. If it exists, $\mathcal{B}$ reports "failure" and aborts; otherwise, $\mathcal{B}$ chooses a random number $r_1 \in Z_q^*$, records $((g^v)^{x_i r}||2, g^{r_1}, r_1, x_i r)$ in Table T_h, and outputs $(r, (g^v)^{x_i r}, (g^{1/v})^{r_1/x_j})$.
  Note that $pk_i = (g^v)^{x_i}$ and $pk_j = (g^v)^{x_j}$.

With the similar analysis in the external security, $\mathcal{B}$ solves the eCDH problem with a non-negligible probability.

Delegatee Security: Compared to the limited proxy, $\mathcal{B}$ needs to change verification keys oracle $\mathcal{O}_{pk}$, signature oracle $\mathcal{O}_s$, and re-signature key generation oracle $\mathcal{O}_{rk}$ as follows.

- *Verification keys oracle $\mathcal{O}_{pk}$:* For the delegatee, set the verification key as $(g^v)^{x_0}$, and for all other users g^{x_i}, where x_i's $(i = 0, \cdots, n)$ are random numbers from Z_q^*.
- *Signature oracle $\mathcal{O}_s$:* On input (pk_0, m_i), $\mathcal{B}$ performs as the same as that in $\mathcal{O}_s$ with input in (pk_t, m_i) in the external security, where pk_0 is treated as pk_t.

- *Re-signature key generation oracle $\mathcal{O}_{rk}$:* On input (pk_i, pk_j), where $pk_j \neq pk_0$, $\mathcal{B}$ performs as the same as that in the real execution since it knows x_j such that $pk_j = g^{x_j}$.

With the similar analysis in the external security, $\mathcal{B}$ solves the eCDH problem with a non-negligible probability.

Delegator Security: Compared to the limited proxy, $\mathcal{B}$ needs to change verification keys oracle $\mathcal{O}_{pk}$, signature oracle $\mathcal{O}_s$, re-signature key generation oracle $\mathcal{O}_{reky}$ as follows.

- *Verification keys oracle $\mathcal{O}_{pk}$:* For the delegator, set the verification key as $(g^v)^{x_0}$, and for all other users g^{x_i}, where x_i's $(i = 0, \cdots, n)$ are random numbers from Z_q^*.
- *Signature oracle $\mathcal{O}_s$:* On input (pk_0, m_i), $\mathcal{B}$ performs as the same as that in $\mathcal{O}_s$ with input in (pk_t, m_i) in the external security, where pk_0 is treated as pk_t.
- *Re-signature key generation oracle $\mathcal{O}_{rk}$:* On input (pk_i, pk_j),
  - if $pk_j \neq pk_0$, then $\mathcal{B}$ gets x_j from Table T_{pk}, and uses x_j to run $\texttt{Rekey}(pk_i, pk_j)$. At last, $\mathcal{B}$ outputs the result from $\texttt{ReKey}$.
  - if $pk_j = pk_0$, then $\mathcal{B}$ chooses a random number $r \in Z_q^*$, and checks whether $(pk_i^r||2, *, *, *)$ exists in Table T_h. If it exists, $\mathcal{B}$ reports "failure" and aborts; otherwise, $\mathcal{B}$ chooses a random number $r_1 \in Z_q^*$, records $(pk_i^r||2, g^{r_1}, r_1, \perp)$ in Table T_h, and outputs $(r, pk_i^r, (g^{1/v})^{r_1/x_0})$.

With the similar analysis in the external security, $\mathcal{B}$ solves the eCDH problem with a non-negligible probability. Note that in this case, the forgery σ^* is an owner-type signature.

Hence, we get this theorem. □

3.6 An Attack on Scheme S_{us}

Now, we consider the following case: Alice → Proxy → Bob. First, Alice can produce an owner-type signature on m: $\sigma_a = (H(m||0)^r, g^r, H(g^r||1)^a)$, where she knows the value of r. Then Proxy can transform σ_a into Bob's signature $\sigma_b = (H(m||0)^r, g^r, (H(g^r||1)^a)^{rk^{(1)}_{a\to b}}, rk^{(2)}_{a\to b}, rk^{(3)}_{a\to b})$. In this case, Alice can generate signatures on any message, simply by changing m to m', since she knows the value of r. This shows that scheme S_{us} is insecure. Hence, the AH model is not suitable for `UPRS-prk`. Note that most of the existing unidirectional proxy re-signature schemes are `UPRS-prk` schemes; hence, it is desired to propose a new security model to solve this problem.

Remark 3. The scheme S_{us} *cannot* be considered as a unidirectional PRS scheme with public re-signature key, since it holds external security which classifies the unidirectional PRS schemes with private re-signature key or not [5,12].

Remark 4. The main reason why scheme S_{us} is insecure is that the re-sign algorithm does not affect the value containing the message m in the owner-type signature. However, the schemes in [5,12] do not have this flaw.

4 The Proposed Security Model for UPRS-prk

In this section, we propose a new security model for UPRS-prk, which covers the attack in Section 3.6. Due to its simplicity, it is easy to verify its completeness.

Before giving our security model, we would first define several terms.

1. If user A delegates his signing rights to user B via a proxy P, then both user A and user B are said to be in a *delegation chain*, denoted as (B,A). User B is called user A's *delegation predecessor*. The combination of the proxy and a user, either the delegatee B or the delegator A, is called a *delegation pair*. Therefore user A and proxy P is a *delegation pair*. So is user B and proxy P.
2. If one of parties in a delegation pair is corrupted, then the delegation pair is *corrupted*; otherwise, it is *uncorrupted*.
3. A user can be treated as the smallest *delegation chain*.
4. If two users A and B are in a delegation chain and B is A's delegation predecessor, then B's signature can be transformed by a proxy or proxies into A's signature.
5. A delegation chain is its own *subchain*.
6. *(Only for multi-use UPRS-prk.)* If user A delegates his signing rights to user B via a proxy P, and user B delegates his signing rights to user C via a proxy P', then user A and user C are said to be in a *delegation chain* too. User C is also called user A's *delegation predecessor*. In this case, users A, B, C are in a delegation chain (C,B,A). The delegation chains (B,A) and (C,B) are *delegation subchains* of the delegation chain (C,B,A). The delegation chain (C,B,A) can be extended if C delegates his signing rights to another user via another proxy.

Existential Unforgeability for UPRS-prk. The existential unforgeability for UPRS-prk is defined by the following adaptively chosen-message attack game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Note that we work in a static mode, that is, before the game starts, the adversary should decide which users and proxies are corrupted, and all the verification keys in the security model are generated by the challenger.

Queries: The adversary adaptively makes a number of different queries to the challenger. Each query can be one of the following.

- *Verification Key query* $\mathcal{O}_{pk}$. On input an index $i \in \{1, \cdots, n\}$ by the adversary[3], the challenger responds by running $\mathtt{KeyGen}(1^k)$ to get a key pair (pk_i, sk_i), and forwards the verification key pk_i to the adversary. At last, the challenger records (pk_i, sk_i) in the list T_K.
- *Signing Key query* $\mathcal{O}_{sk}$. On input a verification key pk_i by the adversary, the challenger responds sk_i which is the associated value with pk_i in the list T_K, if pk_i is corrupted; otherwise, the challenger responds with $\perp$.

[3] We assume that the adversary never inputs number twice. If so, the challenger simply returns the previous value.

- *Re-signature Key query* $\mathcal{O}_{rk}$. On input two verification keys (pk_i, pk_j) $(pk_i \neq pk_j)$ by the adversary, the challenger responds with $\texttt{ReKey}(pk_i, pk_j, sk_j)$, where sk_i is the signing key of pk_i.
- *Signature query* $\mathcal{O}_s$. On input a verification key pk_i, and a message m_i by the adversary, the challenger responds with $\texttt{Sign}(sk_i, m, 1)$, where sk_i is the signing key of pk_i.
- *Re-Signature query* $\mathcal{O}_{rs}$. On input two verification keys pk_i, pk_j $(pk_i \neq pk_j)$, a message m_i, and a signature σ_i at level ℓ by the adversary, the challenger responds with $\texttt{ReSign}(\texttt{ReKey}(pk_i, pk_j, sk_j), pk_i, m_i, \sigma_i, \ell)$, where sk_i is the signing key of pk_i.

Forgery: The adversary output a message m^*, a verification key pk^*, and a signature σ^* at level ℓ^*. The adversary *wins* if the following hold true:

1. $\texttt{Verify}(pk^*, m^*, \sigma^*, \ell^*) = 1$.
2. pk^* is uncorrupted.
3. The adversary has not made a signature query on (pk^*, m^*).
4. The adversary has not made a signature query on (pk', m^*), where pk' is uncorrupted, and there exists such a delegation subchain from pk' to pk^* that does not contain any uncorrupted delegation pair;
5. The adversary has not made a re-signature key query on (pk_i, pk_j), which satisfies all the following conditions:
   - pk_i is corrupted,
   - pk_j is uncorrupted,
   - there exists such a delegation subchain from pk_j to pk^* that does not contain any uncorrupted delegation pair.
6. The adversary has not made a re-signature query on $(pk_i, pk_j, m^*, \sigma_i, *)$, where pk_j is uncorrupted, and there exists such a delegation subchain from pk' to pk^* that does not contain any uncorrupted delegation pair.

We define $\mathbf{Adv}_{\mathcal{A}}$ to be the probability that adversary $\mathcal{A}$ wins in the above game.

Definition 3. *A* `UPRS-prk` *scheme is existentially unforgeable with respect to adaptive chosen message attacks if for all p.p.t. adversaries* $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}$ *is negligible in* k.

Remark 5 (Winning Requirements). The first requirement guarantees that $(pk^*, m^*, \sigma^*, \ell^*)$ is a valid signature. The second requirement guarantees that the adversary cannot trivially obtained a valid owner-type signature by obtaining the signing key. The third requirement guarantees that the adversary cannot trivially obtained a valid owner-type signature by the signature oracle. The fourth and fifth requirements guarantee that the adversary cannot trivially obtained a valid non-owner-type signature by the re-signature key oracle and re-signature oracle, respectively.

Remark 6 (Chosen Key Model). Following the spirit in [12], we can easily extend our security model into the chosen key model [6], where the central authority does not need to verify that the owner of one verification key indeed knows the corresponding signing key. In particular, the challenger is no longer responsible for replying the query that needs the signing key of the corrupted verification key to answer.

Remark 7 (Relationship between the AH Model and the Proposed Model). It is clear to see that scheme S_{us} cannot be proven secure in the proposed model due to the attack in Section 3. Hence, the proposed model is not weaker than the AH model if the treated PRS scheme is a `UPRS-prk` scheme. However, these two models are incomparable if the treated PRS scheme is not a `UPRS-prk` scheme, since our proposed model only deals with `UPRS-prk`.

Remark 8 (Existential Unforgeability for Existing UPRS-prk Schemes). It is interesting that the private re-signature key, unidirectional proxy re-signature schemes in [5,12], which are proven secure in the AH model, are still proven secure in our security model. The main reason is that unlike scheme S_{us}, the re-sign algorithms of these schemes affect the value including m in the owner-type signature. Due to the limited space, we will give these proofs in the full version.

5 Conclusions

In this paper, we pointed out that the AH model proposed in [5] cannot guarantee all desired security requirements for all kinds of unidirectional proxy re-signatures. To show this, we artificially constructed a single-use, private re-signature key, unidirectional proxy re-signature scheme S_{us} which is insecure but proven secure in the AH model. We then proposed a new security model to address the deficiency of the AH model. Fortunately, the private re-signature key, unidirectional proxy re-signature schemes in [5,12] proven secure in the AH model are still proven secure in the new model. Hence, we can still use them in the applications where it demands private re-signature key, unidirectional proxy re-signature.

Acknowledgements

The authors thank the anonymous reviewers for helpful comments. Jun Shao and Peng Liu were supported by ARO STTR "Practical Efficient Graphical Models for Cyber-Security Analysis in Enterprise Networks," AFOSR FA9550-07-1-0527 (MURI), ARO MURI: Computer-aided Human Centric Cyber Situation Awareness, and NSF CNS-0905131. Zhenfu Cao was supported by the National Natural Science Foundation of China (NSFC) under Grant Nos. 60972034, 60970110 and 60773086.

References

1. `http://tdt.sjtu.edu.cn/~jshao/prcbib.htm`
2. The Informed Dialogue about Consumer Acceptability of DRM Solutions in Europe (INDICARE). Consumer Survey on Digital Music and DRM (2005), `http://www.indicare.org/survey`
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In: Internet Society (ISOC): NDSS 2005, pp. 29–43 (2005)

4. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. ACM Transactions on Information and System Security (TISSEC) 9(1), 1–30 (2006)
5. Ateniese, G., Hohenberger, S.: Proxy re-signatures: new definitions, algorithms, and applications. In: ACM CCS 2005, pp. 310–319 (2005)
6. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: ACM CCS 2006, pp. 390–399 (2006)
7. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
8. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
9. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)
10. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
11. Koenen, R., Lacy, J., Mackey, M., Mitchell, S.: The long march to interoperable digital rights management. Proceedings of the IEEE 92(6), 883–897 (2004)
12. Libert, B., Vergnaud, D.: Multi-use unidirectional proxy re-signatures. In: ACM CCS 2008, pp. 511–520 (2008), http://arxiv.org/abs/0802.1113v1
13. Shao, J., Cao, Z., Wang, L., Liang, X.: Proxy re-signature schemes without random oracles. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 197–209. Springer, Heidelberg (2007)
14. Taban, G., Cárdenas, A.A., Gligor, V.D.: Towards a Secure and Interoperable DRM Architecture. In: ACM DRM 2006, pp. 69–78 (2006)
15. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)