Multi-Use Unidirectional Proxy Re-Signatures[*]

Benoît Libert[1] and Damien Vergnaud[2]

[1] Université Catholique de Louvain, Microelectronics Laboratory
Place du Levant, 3 – 1348 Louvain-la-Neuve – Belgium
[2] Ecole Normale Supérieure – C.N.R.S. – I.N.R.I.A.
45, Rue d'Ulm – 75230 Paris CEDEX 05 – France

Abstract. In 1998, Blaze, Bleumer, and Strauss suggested a cryptographic primitive termed *proxy re-signature* in which a proxy transforms a signature computed under Alice's secret key into one from Bob on the same message. The proxy is only semi-trusted in that it cannot learn any signing key or sign arbitrary messages on behalf of Alice or Bob. At CCS 2005, Ateniese and Hohenberger revisited this primitive by providing appropriate security definitions and efficient constructions in the random oracle model. Nonetheless, they left open the problem of constructing a *multi-use unidirectional* scheme where the proxy is only able to translate in one direction and signatures can be re-translated several times.

This paper provides the first steps towards efficiently solving this problem, suggested for the first time 10 years ago, and presents the first *multi-hop unidirectional* proxy re-signature schemes. Although our proposals feature a linear signature size in the number of translations, they are the first multi-use realizations of the primitive that satisfy the requirements of the Ateniese-Hohenberger security model. The first scheme is secure in the random oracle model. Using the same underlying idea, it readily extends into a secure construction in the *standard model* (*i.e.* the security proof of which avoids resorting to the random oracle idealization). Both schemes are computationally efficient but require newly defined Diffie-Hellman-like assumptions in bilinear groups.

Keywords. Digital signatures, multi-use proxy re-cryptography, unidirectionality.

1 Introduction

In 1998, Blaze, Bleumer and Strauss [8] introduced a cryptographic primitive where a semi-trusted proxy is provided with some information that allows turning Alice's signature on a message into Bob's signature on the same message. These *proxy re-signatures* (PRS) – not to be confused with proxy signatures [25] – require that proxies be unable to sign on behalf of Alice or Bob on their own. The recent years saw a renewed interest of the research community in proxy re-cryptography [3, 12, 18–20, 13].

This paper presents the first constructions of *multi-use unidirectional* proxy re-signature wherein the proxy can only translate signatures in one direction and messages can be re-signed a polynomial number of times. Our constructions are efficient and demand new (but falsifiable) Diffie-Hellman-related intractability assumptions in bilinear map groups. One of our contributions is a secure scheme in the standard model (*i.e.* without resorting to the random oracle model).

Related work. Alice – the delegator – can easily designate a proxy translating signatures computed using the secret key of Bob – the delegatee – into one that are valid w.r.t. her public key by storing her secret key at the proxy. Upon receiving Bob's signatures, the proxy can check them and re-sign the message using Alice's private key. The problem with this approach is that the proxy can sign arbitrary messages on behalf of Alice. Proxy re-signatures aim at securely enabling the delegation of signatures without fully trusting the proxy. They are related to proxy

[*] This paper has appeared in *2008 ACM Conference on Computer and Communications Security, CCS 2008*, P. Ning, P. F. Syverson & S. Jha eds., ACM, 2008, pp. 511-520.

signatures [25, 21] in that any PRS can be used to implement a proxy signature mechanism but the converse is not necessarily true.

In 1998, Blaze *et al.* [8] gave the first example of PRS where signing keys remain hidden from the proxy. The primitive was formalized in 2005 by Ateniese and Hohenberger [12] who pinned down useful properties that can be expected from proxy re-signature schemes:

- Unidirectionality: re-signature keys can only be used for delegation in one direction;
- Multi-usability: a message can be re-signed a polynomial number of times;
- Privacy of proxy keys: re-signature keys can be kept secret by honest proxies;
- Transparency: users may not even know that a proxy exists;
- Unlinkability: a re-signature cannot be linked to the signature from which it was generated;
- Key optimality: a user is only required to store a constant amount of secret data;
- Non-interactivity: the delegatee does not act in the delegation process;
- Non-transitivity: proxies cannot re-delegate their re-signing rights.

Blaze *et al.*'s construction is *bidirectional* (*i.e.* the proxy information allows "translating" signatures in either direction) and *multi-use* (*i.e.* the translation of signatures can be performed in sequence and multiple times by distinct proxies without requiring the intervention of signing entities). Unfortunately, Ateniese and Hohenberger [12] pinpointed a flaw in the latter scheme: given a signature/re-signature pair, anyone can deduce the re-signature key that has been used in the delegation (*i.e.* proxy keys are not private). Another issue in [8] is that the proxy and the delegatee can collude to expose the delegator's secret.

To overcome these limitations, Ateniese and Hohenberger [12] proposed two constructions based on bilinear maps. The first one is a multi-use, bidirectional extension of Boneh-Lynn-Shacham (BLS) signatures [11]. Their second scheme is unidirectional (the design of such a scheme was an open problem raised in [8]) but single-use. It involves two different signature algorithms: *first-level* signatures can be translated by the proxy whilst *second-level* signatures (that are obtained by translating first level ones or by signing at level 2) cannot. A slightly less efficient variant was also suggested to ensure the privacy of re-signature keys kept at the proxy. The security of all schemes was analyzed in the random oracle model [7].

Motivations. A number of applications were suggested in [12] to motivate the search for unidirectional systems. One of them was to provide a proof that a certain path was taken in a directed graph: to make sure that a foreign visitor legally entered the country and went through the required checkpoints, U.S. customs only need one public key (the one of the immigration service once the original signature on the e-passport has been translated by an immigration agent). Optionally, the final signature can hide which specific path was chosen and only vouch for the fact that an authorized one was taken. In such a setting, proxy re-signatures are especially interesting when they are multi-use.

Another application was the sharing and the conversion of digital certificates: valid signatures for untrusted public keys can be turned into signatures that verify under already certified keys so as to save the cost of obtaining a new certificate. As exemplified in [12], unidirectional schemes are quite appealing for converting certificates between *ad-hoc* networks: using the public key of network B's certification authority (CA), the CA of network A can non-interactively compute a translation key and set up a proxy converting certificates from network B within its own domain without having to rely on untrusted nodes of B.

As a third application, PRS can be used to implement anonymizable signatures that hide the internal organization of a company. Outgoing documents are first signed by specific employees. Before releasing them to the outside world, a proxy translates signatures into ones that verify under a corporate public key so as to conceal the original issuer's identity and the internal structure of the company.

Our contributions. Ateniese and Hohenberger left as open challenges the design of multi-use unidirectional systems and that of secure schemes in the standard security model. This paper provides solutions to both problems:

- we present a simple and efficient system (built on the short signature put forth by Boneh *et al.* [11]) which is secure in the random oracle model under an appropriate extension of the Diffie-Hellman assumption;
- using an elegant technique due to Waters [31], the scheme is easily modified so as to achieve security in the standard model. To the best of our knowledge, this actually provides the first unidirectional PRS that dispenses with random oracles and thereby improves a recent bidirectional construction [29].

Both proposals additionally preserve the privacy of proxy keys (with an improved efficiency w.r.t. [12] in the case of the first one). They combine almost all of the above properties. As in prior unidirectional schemes, proxies are not completely transparent since signatures have different shapes and lengths across successive levels. The size of our signatures actually grows linearly with the number of past translations: signatures at level ℓ (*i.e.* that have been translated $\ell - i$ times if the original version was signed at level i) consist of about 2ℓ group elements. In spite of this blow-up, we retain important benefits:

- signers may tolerate a limited number (say t) of signature translations for specific messages. Then, if L distinct signature levels are permitted in the global system, users can directly sign messages at level $L - t$.
- the conversion of a ℓ^{th} level signature is indistinguishable from one generated at level $\ell + 1$ by the second signer. The original signer's identity is moreover perfectly hidden and the verifier only needs the new signer's public key.

As a last contribution, we also show how the single-hop restrictions of both schemes can be modified in such a way that one can prove their security in the stronger *plain public key model* (also considered in [4] for different primitives). Prior works on proxy re-cryptography consider security definitions where dishonest parties' public keys are honestly generated and the corresponding secret key is known to the attacker. Relying on the latter assumption requires CAs to ask for a proof of knowledge of the associated private key before certifying a public key. As exemplified in [4], not all security infrastructures do rigorously apply such an advisable practice. To address this issue in our setting, we extend the security definitions of [12] to the *plain public key model* (a.k.a. *chosen-key model*) where the adversary is allowed to choose public keys on behalf of corrupt users (possibly non-uniformly or as a function of honest parties' public keys) without being required to reveal or prove knowledge of the underlying private key. In our model, we are able to construct single-hop unidirectional schemes that are secure in the plain public key model. The practical impact of this result is that users do not have to demonstrate knowledge of their secret upon certification. They must only obtain a standard certificate such as those provided by current PKIs.

Organization. In the forthcoming sections, we recall the syntax of unidirectional PRS schemes and the security model in section 2. Section 3 explains which algorithmic assumptions we need. Section 4 describes our random-oracle-using scheme. In section 5, we detail how to get rid of the random oracle idealization. Section 6 then suggests single-hop constructions in the chosen-key model.

2 Model and Security Notions

We first recall the syntactic definition of unidirectional PRS schemes from [12].

Definition 1 (Proxy Re-Signatures). *A (unidirectional) proxy re-signature (PRS) scheme for N signers and L levels (where N and L are both polynomial in the security parameter λ) is a tuple of (possibly randomized) algorithms* $(\mathsf{Global\text{-}Setup}, \mathsf{Keygen}, \mathsf{ReKeygen}, \mathsf{Sign}, \mathsf{Re\text{-}Sign}, \mathsf{Verify})$ *where:*

$\mathsf{Global\text{-}Setup}(\lambda)$**:** *is a randomized algorithm (possibly run by a trusted party) that takes as input a security parameter λ and produces system-wide public parameters* cp.

$\mathsf{Keygen}(\mathsf{cp})$**:** *is a probabilistic algorithm that, on input of public parameters* cp*, outputs a signer's private/public key pair (sk, pk).*

$\mathsf{ReKeygen}(\mathsf{cp}, pk_i, sk_j)$**:** *on input of public parameters* cp*, the public key pk_i of signer i and signer j's private key sk_j, this (possibly randomized but ideally non-interactive) algorithm outputs a re-signature key R_{ij} that allows turning i's signatures into signatures in the name of j.*

$\mathsf{Sign}(\mathsf{cp}, \ell, sk_i, m)$**:** *on input of public parameters* cp*, a message m, a private key sk_i and an integer $\ell \in \{1, \ldots, L\}$, this (possibly probabilistic) algorithm outputs a signature σ on behalf of signer i at level ℓ.*

$\mathsf{Re\text{-}Sign}(\mathsf{cp}, \ell, m, \sigma, R_{ij}, pk_i, pk_j)$**:** *given common parameters* cp*, a level $\ell < L$ signature σ from signer $i \in \{1, \ldots, N\}$ and a re-signature key R_{ij}, this (possibly randomized) algorithm first checks that σ is valid w.r.t pk_i. If yes, it outputs a signature σ' that verifies at level $\ell + 1$ under the public key pk_j.*

$\mathsf{Verify}(\mathsf{cp}, \ell, m, \sigma, pk_i)$**:** *given public parameters* cp*, an integer $\ell \in \{1, \ldots, L\}$, a message m, an alleged signature σ and a public key pk_i, this deterministic algorithm outputs 0 or 1.*

For all security parameters $\lambda \in \mathbb{N}$ and public parameters cp *output by* $\mathsf{Global\text{-}Setup}(\lambda)$*, for all couples of private/public key pairs (sk_i, pk_i), (sk_j, pk_j) produced by* $\mathsf{Keygen}(\mathsf{cp})$*, for any $\ell \in \{1, \ldots, L\}$ and message m, we should have*

$$\mathsf{Verify}(\mathsf{cp}, \ell, m, \mathsf{Sign}(\mathsf{cp}, \ell, sk_i, m), pk_i) = 1;$$
$$\mathsf{Verify}(\mathsf{cp}, \ell + 1, m, \sigma, pk_j) = 1.$$

whenever $\sigma = \mathsf{ReSign}(\mathsf{cp}, \ell, m, \mathsf{Sign}(\mathsf{cp}, \ell, sk_i, m), R_{ij})$ *and* $R_{ij} = \mathsf{ReKeygen}(\mathsf{cp}, pk_i, sk_j)$.

To lighten notations, we sometimes omit to explicitly include public parameters cp that are part of the input of some of the above algorithms.

The security model of [12] considers the following two orthogonal notions termed *external* and *insider security.*

External security: is the security against adversaries outside the system (that differ from the proxy and delegation partners). This notion demands that the next probability be a negligible function of the security parameter λ:

$$\Pr[\ \{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i \in [1,N]}, (i^\star, L, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(.), \mathcal{O}_{Resign}(.)}(\{pk_i\}_{i \in [1,N]}) : \\ \mathsf{Verify}(L, m^\star, \sigma^\star, pk_{i^\star}) \wedge (i^\star, m^\star) \notin Q]$$

where $\mathcal{O}_{Sign}(.)$ is an oracle taking as input a message and an index $i \in \{1, \ldots, N\}$ to return a 1st-level signature $\sigma \leftarrow \mathsf{Sign}(1, sk_i, m)$; $\mathcal{O}_{Resign}(.)$ takes indices $i, j \in \{1, \ldots, N\}$ and a ℓ^{th}-level signature σ to output $\sigma' \leftarrow \mathsf{Re\text{-}Sign}(\ell, m, \sigma, \mathsf{ReKeygen}(pk_i, sk_j), pk_i, pk_j)$; and Q denotes the set of (signer,message) pairs (i, m) queried to $\mathcal{O}_{Sign}(.)$ or such that a tuple $(?, j, i, m)$, with $j \in \{1, \ldots, N\}$, was queried to $\mathcal{O}_{Resign}(.)$. This notion only makes sense if re-signing keys are kept private by the proxy.

In our setting, the translation of a ℓ^{th}-level signature is perfectly indistinguishable from a signature produced by the delegator at level $\ell + 1$. Therefore, we can always simulate the $\mathcal{O}_{Resign}(.)$ oracle by publicly "sending" outputs of $\mathcal{O}_{Sign}(.)$ to the next levels. For the sake of generality, we nevertheless leave $\mathcal{O}_{Resign}(.)$ in the definition.

Internal security: The second security notion considered in [12] strives to protect users against dishonest proxies and colluding delegation partners. Three security guarantees should be ensured.

1. **Limited Proxy security:** this notion captures the proxy's inability to sign messages on behalf of the delegatee or to create signatures for the delegator unless messages were first signed by one of the latter's delegatees. Formally, we consider a game where adversaries have all re-signing keys but are denied access to signers' private keys. The following probability should be negligible:

$$\begin{aligned}\Pr\big[\ &\{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i\in[1,N]}, \{R_{ij} \leftarrow \mathsf{ReKeygen}(pk_i, sk_j)\}_{i,j\in[1,N]},\\ &(i^\star, L, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(.,.)}\big(\{pk_i\}_{i\in[1,N]}, \{R_{ij}\}_{i,j\in[1,N]}\big) :\\ &\mathsf{Verify}(L, m^\star, \sigma^\star, pk_{i^\star}) \wedge m^\star \notin Q\big]\end{aligned}$$

where $\mathcal{O}_{Sign}(.,.)$ is an oracle taking as input a message and an index $i \in \{1, \ldots, N\}$ to return a first level signature $\sigma \leftarrow \mathsf{Sign}(1, sk_i, m)$ and Q stands for the set of messages m queried to the signing oracle.

2. **Delegatee Security:** informally, this notion protects the delegatee from a colluding delegator and proxy. Namely, the delegatee is assigned the index 0. The adversary is provided with an oracle returning first level signatures on behalf of 0. Knowing corrupt users' private keys, she can compute re-signature keys $\{R_{ij}\}_{i\in\{0,\ldots,N\},j\in\{1,\ldots,N\}}$ on her own[3] from pk_i and sk_j, with $j \neq 0$. Obviously, she is not granted access to R_{i0} for any $i \neq 0$. Her probability of success

$$\begin{aligned}\Pr\big[\ &\{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i\in[0,N]}, (L, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(0,.)}\big(pk_0, \{pk_i, sk_i\}_{i\in[1,N]}\big) :\\ &\mathsf{Verify}(L, m^\star, \sigma^\star, pk_0) \wedge m^\star \notin Q\ \big],\end{aligned}$$

where Q is the set of messages queried to $\mathcal{O}_{Sign}(0, .)$, should be negligible.

3. **Delegator Security:** this notion captures that a collusion between the delegatee and the proxy should be harmless for the honest delegator. More precisely, we consider a target delegator with index 0. The adversary is given private keys of all other signers $i \in \{1, \ldots, N\}$ as well as *all* re-signature keys including R_{i0} and R_{0i} for $i \in \{1, \ldots, N\}$. A signing oracle $\mathcal{O}_{Sign}(0, .)$ also provides her with first level signatures for 0. Yet, the following probability should be negligible,

$$\begin{aligned}\Pr\big[\ &\{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i\in[0,N]}, \{R_{ij} \leftarrow \mathsf{ReKeygen}(pk_i, sk_j)\}_{i,j\in[0,N]},\\ &(1, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(0,.)}\big(pk_0, \{pk_i, sk_i\}_{i\in[1,N]}, \{R_{ij}\}_{i,j\in[0,N]},\big) :\\ &\mathsf{Verify}(1, m^\star, \sigma^\star, pk_0) \wedge m^\star \notin Q\big],\end{aligned}$$

meaning she has little chance of framing user 0 at the first level.

[3] This is true in in non-interactive schemes, which we are focusing on. In the general case, those keys should be generated by the challenger and explicitly provided as input to the adversary.

An important difference between external and limited proxy security should be underlined. In the former, the attacker is allowed to obtain signatures on the target message $m^\star$ for signers other than $i^\star$. In the latter, the target message cannot be queried for signature at all (knowing all proxy keys, the attacker would trivially win the game otherwise).

Chosen-Key Model Security. As in other papers on proxy re-cryptography [3, 13], the above model assumes that users only publicize a public key if they hold the underlying private key. This actually amounts to use a trusted key generation model or the so-called *knowledge-of-secret-key* model (KOSK), introduced in [9], that demands attackers to reveal the associated private key whenever they create a public key for themselves. This model (sometimes referred to as the *registered key model*) mirrors the fact that, in a PKI, users should prove knowledge of their private key upon certification of their public key.

As argued by Bellare and Neven in a different context [4], relying on the registered key model can be quite burdensome in real world applications if one is willing to actually implement the requirements of that model. Although some kinds of proof of private key possession [27] are implemented by VeriSign and other security infrastructures, they are far from sufficing to satisfy assumptions that are implicitly made by the KOSK model. To do so, CAs should implement complex proofs of knowledge that allow for the online extraction of adversarial secrets so as to remain secure in a concurrent setting like the Internet, where many users may be willing to register at the same time. Hence, whenever it is possible, one should preferably work in a model called *chosen-key model* (a.k.a. *plain public key model*) that leaves adversaries choose their public key as they like (possibly as a function of honest parties' public keys and without having to know or reveal the underlying secret whatsoever).

If we place ourselves in the chosen-key model, the notions of external security and limited proxy security are not altered as they do not involve corrupt users. On the other hand, we need to recast the definitions of delegatee and delegator security and take adversarially-generated public keys into account. As to the delegatee security, the only modification is that the adversary is challenged on a single public key. No other change is needed since $\mathcal{A}$ can generate re-signature keys on her own. In the notion of delegator security, $\mathcal{A}$ is also challenged on a single public key pk_0 for which she is granted access to a first level signing oracle. In addition, we introduce a delegation oracle $\mathcal{O}_{dlg}(.)$ that delegates on behalf of user 0. When queried on a public key pk_i supplied by the adversary, $\mathcal{O}_{dlg}(.)$ responds with $R_{i0} = \mathsf{ReKeygen}(pk_i, sk_0)$.

We stress that we are not claiming that the schemes of [12] are insecure in such a model. However, their security is not guaranteed any longer with currently known security proofs. In section 6, we will explain how to simply modify the single-hop versions of our schemes so as to prove them secure without making the KOSK assumption.

3 Bilinear Maps and Complexity Assumptions

Bilinear groups. Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear map groups* if there is an efficiently computable mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with these properties:

1. bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$;
2. non-degeneracy: $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

Flexible Diffie-Hellman problems. Our signatures rely on new generalizations of the Computational Diffie-Hellman (CDH) problem which is to compute g^{ab} given (g^a, g^b) in a group $\mathbb{G} = \langle g \rangle$. To motivate them, let us first recall the definition of the *2-out-of-3 Diffie-Hellman* problem [22].

Definition 2. *In a prime order group $\mathbb{G}$, the* **2-out-of-3 Diffie-Hellman** *problem (2-3-CDH) is, given (g, g^a, g^b), to find a pair $(C, C^{ab}) \in \mathbb{G} \times \mathbb{G}$ with $C \neq 1_{\mathbb{G}}$.*

We introduce a potentially harder version of this problem that we call 1-Flexible Diffie-Hellman problem:

Definition 3. *The* **1-Flexible Diffie-Hellman** *problem (1-FlexDH) is, given $(g, A = g^a, B = g^b) \in \mathbb{G}^3$, to find a triple $(C, C^a, C^{ab}) \in (\mathbb{G} \backslash \{1_{\mathbb{G}}\})^3$.*

We shall rely on a relaxed variant of this problem where more flexibility is permitted in the choice of the base C for the Diffie-Hellman computation.

Definition 4. *The* **ℓ-Flexible Diffie-Hellman** *problem (ℓ-FlexDH) is, given $(g, A = g^a, B = g^b) \in \mathbb{G}^3$, to find a $(2\ell + 1)$-uple*

$$(C_1, \ldots, C_\ell, D_1^a, \ldots, D_\ell^a, D_\ell^{ab}) \in \mathbb{G}^{2\ell+1}$$

where $\log_g(D_j) = \prod_{i=1}^{j} \log_g(C_i) \neq 0$ for $j \in \{1, \ldots \ell\}$.

A given instance has many publicly verifiable solutions: a candidate $2\ell + 1$-tuple

$$(C_1, \ldots, C_\ell, D'_1, \ldots, D'_\ell, T)$$

is acceptable if $e(C_1, A) = e(D'_1, g)$, $e(D'_j, g) = e(D'_{j-1}, C_j)$ for $j = 2, \ldots, \ell$ and $e(D'_\ell, B) = e(T, g)$. The ℓ-FlexDH assumption is thus falsifiable according to Naor's classification [26].

In generic groups, the general intractability result given by theorem 1 of [22] by Kunz-Jacques and Pointcheval implies the generic hardness of ℓ-FlexDH. Section 8 gives an adaptation of this result in generic *bilinear* groups.

Remark The *knowledge-of-exponent assumption* (KEA1) [5] was introduced by Damgård [15]. Roughly speaking, it captures the intuition that any algorithm $\mathcal{A}$ which, given elements (g, g^x) in $\mathbb{G}^2$, computes a pair $(h, h^x) \in \mathbb{G}^2$ must "know" $\log_g(h)$. Hence, it must be feasible to recover the latter value using $\mathcal{A}$'s random coins. In [6], Bellare and Palacio defined a slightly stronger variant (dubbed DHK1 as a shorthand for "Diffie-Hellman knowledge") of this assumption. DHK1 essentially says that, given a pair (g, g^x), for any adversary $\mathcal{A}$ that outputs pairs (h_i, h_i^x), there exists an extractor that can always recover $\log_g(h_i)$ using $\mathcal{A}$'s random coins. The latter is allowed to query the extractor on polynomially-many pairs (h_i, h_i^x). For each query, $\mathcal{A}$ first obtains $\log_g(h_i)$ from the extractor before issuing the next query. Under DHK1, the intractability of the ℓ-Flexible Diffie-Hellman problem is easily seen to boil down to the Diffie-Hellman assumption. Given a pair (g, g^a), a polynomial adversary that outputs $(C_1, D_1^a) = (C_1, C_1^a)$ necessarily "knows" $t_1 = \log_g C_1$ and thus also $(C_2, C_2^a) = (C_2, (D_2^a)^{1/t_1})$ as well as $t_2 = \log_g C_2$, which in turn successively yields logarithms of $C_3, \ldots, C_\ell$. Although DHK1-like assumptions are inherently non-falsifiable, they hold in generic groups [16, 1] and our results can be seen as resting on the combination CDH+DHK1.

MODIFIED DIFFIE-HELLMAN PROBLEM. The second assumption that we need is that the CDH problem (g^a, g^b) remains hard even when $g^{(a^2)}$ is available.

Definition 5. *The* **modified Computational Diffie-Hellman** *problem (mCDH) is, given $(g, g^a, g^{(a^2)}, g^b) \in \mathbb{G}^4$, to compute $g^{ab} \in \mathbb{G}$.*

In fact, we use an equivalent formulation of the problem which is to find h^{xy} given $(h, h^x, h^{1/x}, h^y)$ (where we set $g = h^{1/x}$, $x = a$, $y = b/a$).

4 A Multi-Hop Scheme in the Random Oracle Model

To provide a better intuition of the underlying idea of our scheme, we first describe its single-hop version before extending it into a multi-hop system.

Our approach slightly differs from the one in [12] where signers have a "strong" secret and a "weak" secret that are respectively used to produce first and second level signatures. In our scheme, users have a single secret but first and second level signatures retain different shapes. Another difference is that our re-signature algorithm is probabilistic.

We exploit the idea that, given $g^b \in \mathbb{G} = \langle g \rangle$ for some $b \in \mathbb{Z}$, one can hardly generate a Diffie-Hellman triple (g^a, g^b, g^{ab}) without knowing the corresponding exponent a [15]. A valid BLS signature [11] $(\sigma = H(m)^x, X = g^x)$ can be blinded into $(\sigma_1', \sigma_2') = (\sigma^t, X^t)$ using a random exponent t. An extra element g^t then serves as evidence that (σ_1', σ_2') actually hides a valid pair. This technique can be iterated several times by adding two group elements at each step. To translate signatures from signer i to signer j, the key idea is to have the proxy perform an appropriate change of variable involving the translation key during the blinding.

The scheme is obviously not strongly unforgeable in the sense of [2] (since all but first level signatures can be publicly re-randomized) but this "malleability" of signatures is not a weakness whatsoever. It even turns out to be a desirable feature allowing for the unlinkability of translated signatures w.r.t. original ones.

4.1 The Single Hop Version

In this scheme, signers' public keys consist of a single group element $X = g^x \in \mathbb{G}$. Their well-formedness is thus efficiently verifiable by the certification authority that just has to check their membership in $\mathbb{G}$. This already improves [12] where public keys $(X_1, X_2) = (g^x, h^{1/x}) \in \mathbb{G}^2$ (g and h being common parameters) must be validated by testing whether $e(X_1, X_2) = e(g, h)$.

Global-setup(λ)**:** this algorithm chooses bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. A generator $g \in \mathbb{G}$ and a hash function $H : \{0,1\}^* \to \mathbb{G}$ (modeled as a random oracle in the security proof) are also chosen. Public parameters only consist of $\mathsf{cp} := \{\mathbb{G}, \mathbb{G}_T, g, H\}$.

Keygen(λ)**:** user i's public key is set as $X_i = g^{x_i}$ for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$.

ReKeygen(x_j, X_i)**:** this algorithm outputs the re-signature key $R_{ij} = X_i^{1/x_j} = g^{x_i/x_j}$ which allows turning signatures from i into signatures from j.

Sign$(1, x_i, m)$**:** to sign $m \in \{0,1\}^*$ at level 1, compute $\sigma^{(1)} = H(m)^{x_i} \in \mathbb{G}$.

Sign$(2, x_i, m)$**:** to sign $m \in \{0,1\}^*$ at level 2, choose $t \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2) = (H(m)^{x_i t}, X_i^t, g^t). \tag{1}$$

Re-Sign$(1, m, \sigma^{(1)}, R_{ij}, X_i, X_j)$**:** on input of $m \in \{0,1\}^*$, the re-signature key $R_{ij} = g^{x_i/x_j}$, a signature $\sigma^{(1)} \in \mathbb{G}$ and public keys X_i, X_j, check the validity of $\sigma^{(1)}$ w.r.t signer i by testing $e(\sigma^{(1)}, g) = e(H(m), X_i)$. If valid, $\sigma^{(1)}$ is turned into a signature on behalf of j by choosing $t \xleftarrow{R} \mathbb{Z}_p^*$ and computing

$$\sigma^{(2)} = (\sigma_0', \sigma_1', \sigma_2') = ({\sigma^{(1)}}^t, X_i^t, R_{ij}^t) = (H(m)^{x_i t}, X_i^t, g^{t x_i/x_j})$$

If we set $\tilde{t} = t x_i / x_j$, we have

$$\sigma^{(2)} = (\sigma_0', \sigma_1', \sigma_2') = (H(m)^{x_j \tilde{t}}, X_j^{\tilde{t}}, g^{\tilde{t}}). \tag{2}$$

Verify$(1, m, \sigma^{(1)}, X_i)$**:** accept $\sigma^{(1)}$ if $e(\sigma^{(1)}, g) = e(H(m), X_i)$.

Verify$(2, m, \sigma^{(2)}, X_i)$**:** a 2nd level signature $\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2)$ is accepted for the public key X_i if the following conditions are true.

$$e(\sigma_0, g) = e(\sigma_1, H(m)) \qquad e(\sigma_1, g) = e(X_i, \sigma_2)$$

Relations (1) and (2) show that translated signatures have exactly the same distribution as signatures directly produced by signers at level 2.

In comparison with the only known unidirectional PRS with private re-signing keys (suggested in section 3.4.2 of [12]), this one features shorter second level signatures that must include a Schnorr-like [28] proof of knowledge in addition to 3 group elements in [12]. On the other hand, signatures of [12] are strongly unforgeable unlike ours.

It is also worth mentioning that the above scheme only requires the 1-Flexible Diffie-Hellman assumption which is more classical than the general ℓ-FlexDH.

4.2 How to Obtain Multiple Hops

The above construction can be scaled up into a multi-hop PRS scheme if we iteratively apply the same idea several times. To prevent the linkability of signatures between successive levels $\ell+1$ and $\ell+2$, the re-signature algorithm performs a re-randomization using random exponents $r_1, \ldots, r_\ell$.

Sign$(\ell+1, x_i, m)$**:** to sign $m \in \{0,1\}^*$ at the $(\ell+1)^{\text{th}}$ level, user i chooses $(t_1, \ldots, t_\ell) \overset{R}{\leftarrow} (\mathbb{Z}_p^*)^\ell$ and outputs $\sigma^{(\ell+1)} = (\sigma_0, \ldots, \sigma_{2\ell}) \in \mathbb{G}^{2\ell+1}$ where

$$\begin{cases} \sigma_0 = H(m)^{x_i t_1 \cdots t_\ell} \\ \sigma_k = g^{x_i t_1 \cdots t_{\ell+1-k}} & \text{for } k \in \{1, \ldots, \ell\} \\ \sigma_k = g^{t_{k-\ell}} & \text{for } k \in \{\ell+1, \ldots, 2\ell\}. \end{cases}$$

Re-Sign$(\ell+1, m, \sigma^{(\ell+1)}, R_{ij}, X_i, X_j)$**:** on input of a message $m \in \{0,1\}^*$, the re-signature key $R_{ij} = g^{x_i/x_j}$, a valid $(\ell+1)^{\text{th}}$-level signature

$$\begin{aligned} \sigma^{(\ell+1)} &= (\sigma_0, \ldots, \sigma_{2\ell}) \\ &= (H(m)^{x_i t_1 \cdots t_\ell}, g^{x_i t_1 \cdots t_\ell}, g^{x_i t_1 \cdots t_{\ell-1}}, \ldots, g^{x_i t_1}, g^{t_1}, \ldots, g^{t_\ell}) \in \mathbb{G}^{2\ell+1} \end{aligned}$$

and public keys X_i, X_j, check the validity of $\sigma^{(\ell+1)}$ under X_i. If valid, it is turned into a $(\ell+2)^{\text{th}}$-level signature on behalf of j by drawing $(r_0, \ldots, r_\ell) \overset{R}{\leftarrow} (\mathbb{Z}_p^*)^{\ell+1}$ and computing $\sigma^{(\ell+2)} = (\sigma'_0, \ldots, \sigma'_{2\ell+2}) \in \mathbb{G}^{2\ell+3}$ where

$$\begin{cases} \sigma'_0 = \sigma_0^{r_0 \cdots r_\ell} \\ \sigma'_k = \sigma_k^{r_0 \cdots r_{\ell+1-k}} & \text{for } k \in \{1, \ldots, \ell\} \\ \sigma'_{\ell+1} = X_i^{r_0} \\ \sigma'_{\ell+2} = R_{ij}^{r_0} \\ \sigma'_k = \sigma_{k-2}^{r_{k-\ell-2}} & \text{for } k \in \{\ell+3, \ldots, 2\ell+2\}. \end{cases}$$

If we define $\tilde{t}_0 = r_0 x_i / x_j$ and $\tilde{t}_k = r_k t_k$ for $k = 1, \ldots, \ell$, we observe that

$$\sigma^{(\ell+2)} = (H(m)^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_\ell}, g^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_\ell}, g^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_{\ell-1}}, \ldots, g^{x_j \tilde{t}_0}, g^{\tilde{t}_0}, \ldots, g^{\tilde{t}_\ell}) \in \mathbb{G}^{2\ell+3}$$

Verify$(\ell+1, m, \sigma^{(\ell+1)}, X_i)$**:** at level $(\ell+1)$, the validity of $\sigma^{(\ell+1)} = (\sigma_0, \ldots, \sigma_{2\ell}) \in \mathbb{G}^{2\ell+1}$ is checked by testing if these equalities simultaneously hold:

$$\begin{aligned} & e(\sigma_0, g) = e(H(m), \sigma_1), \\ & e(\sigma_\ell, g) = e(X_i, \sigma_{\ell+1}) \\ & e(\sigma_k, g) = e(\sigma_{k+1}, \sigma_{2\ell-k+1}) \text{ for } k \in \{1, \ldots, \ell-1\} \end{aligned}$$

We note that the speed of the verification algorithm can be increased by computing a product of $O(\ell)$ pairings, which is significantly faster than $O(\ell)$ independent pairing calculations [17]. The idea is to choose $\omega_0, \dots, \omega_\ell \overset{R}{\leftarrow} \mathbb{Z}_p^*$ at random and check whether

$$e\big(g, \prod_{k=0}^{\ell} \sigma_k^{\omega_k}\big) = e(H(m), \sigma_1{}^{\omega_0}) \cdot e(X_i, \sigma_{\ell+1}^{\omega_\ell}) \cdot \prod_{k=1}^{\ell-1} e(\sigma_{k+1}, \sigma_{2\ell-k+1}^{\omega_k}).$$

With high probability, invalid signatures fail to satisfy the above randomized verification algorithm.

4.3 Security

Theorem 1. *The L-level scheme is a secure unidirectional proxy re-signature under the* $(L-1)$-*FlexDH and mCDH assumptions in the random oracle model.*

Proof. We first prove security against dishonest proxies.

Limited proxy security From an adversary $\mathcal{A}_1$ with advantage ε, we can construct an algorithm $\mathcal{B}_1$ that solves a $(L-1)$-FlexDH instance $(g, A = g^a, B = g^b)$ with probability $O(\varepsilon/q_s)$, where q_s is the number of signing queries.

System parameters: $\mathcal{A}_1$ is challenged on public parameters $\{\mathbb{G}, \mathbb{G}_T, g, \mathcal{O}_H\}$ where $\mathcal{O}_H$ is the random oracle controlled by the simulator $\mathcal{B}_1$.

Public key generation: when $\mathcal{A}_1$ asks for the creation of user $i \in \{1, \dots, N\}$, $\mathcal{B}_1$ responds with a newly generated public key $X_i = A^{x_i} = g^{ax_i}$, for a random $x_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$, which virtually defines user i's private key as ax_i. For all pairs (i, j), re-signature keys R_{ij} are calculated as $R_{ij} = g^{x_i/x_j} = g^{ax_i/ax_j}$.

Oracle queries: $\mathcal{A}_1$'s queries are tackled with as follows. Following a well-known technique due to Coron [14], a binary coin $c \in \{0, 1\}$ with expected value $1 - \zeta \in [0, 1]$ decides whether $\mathcal{B}_1$ introduces the challenge in the output of the random oracle or an element of known signature. For the optimal value of ζ, this introduces the loss factor $O(q_s)$ in the success probability.

- *Random oracle queries*: to answer these queries, $\mathcal{B}_1$ maintains a list (referred to as the H-List) of tuples (m, h, μ, c) as follows:
  1. If the query m already appears in the H-List, then $\mathcal{B}_1$ returns h;
  2. Otherwise, $\mathcal{B}_1$ generates a random bit c such that $\Pr[c = 0] = \zeta$;
  3. It picks $\mu \overset{R}{\leftarrow} \mathbb{Z}_p^*$ at random and computes $h = g^\mu$ if $c = 0$ and $h = B^\mu$ otherwise;
  4. It adds the 4-uple (m, h, μ, c) to the H-List and returns h as the random oracle output.
- *Signing queries*: when a signature of signer i is queried for a message m, $\mathcal{B}_1$ runs the random oracle to obtain the 4-uple (m, h, μ, c) contained in the H-List. If $c = 1$ then $\mathcal{B}_1$ reports failure and aborts. Otherwise, the algorithm $\mathcal{B}_1$ returns $h^{x_i a} = A^{x_i \mu}$ as a valid signature on m.

After a number of queries, $\mathcal{A}_1$ comes up with a message $m^\star$, that was never queried for signature for any signer, an index $i^\star \in \{1, \dots, N\}$ and a L^{th} level forgery $\sigma^{\star(L)} = (\sigma_0^\star, \dots, \sigma_{2L-2}{}^\star) \in \mathbb{G}^{2L-1}$. At this stage, $\mathcal{B}_1$ runs the random oracle to obtain the 4-uple $(m^\star, h^\star, \mu^\star, c^\star)$ contained in the H-List and fails if $c^\star = 0$. Otherwise, if $\sigma^{\star(L)}$ is valid, it may be written

$$(\sigma_0^\star, \dots, \sigma_{2L-2}{}^\star) = \Big(B^{\mu^\star x_{i^\star} a t_1 \dots t_{L-1}}, A^{t_1, \dots t_{L-1}}, \dots, A^{t_1}, g^{t_1}, \dots, g^{t_{L-1}}\Big)$$

which provides $\mathcal{B}_1$ with a valid tuple

$$(C_1, \ldots, C_{L-1}, D_1^a, \ldots, D_{L-1}^a, D_{L-1}^{ab}),$$

where $D_{L-1}^{ab} = \sigma_0^{\star 1/\mu^\star x_{i^\star}}$, so that $\log_g(D_j) = \prod_{i=1}^{j} \log_g(C_i)$ for $j \in \{1, \ldots, L-1\}$. A similar analysis to [14, 11] gives the announced bound on $\mathcal{B}_1$'s advantage if the optimal probability $\zeta = q_s/(q_s + 1)$ is used when answering hash queries.

Delegatee security We also attack the $(L-1)$-FlexDH assumption using a delegatee security adversary $\mathcal{A}_2$. Given an input pair $(A = g^a, B = g^b)$, the simulator $\mathcal{B}_2$ proceeds as $\mathcal{B}_1$ did in the proof of limited proxy security.

System parameters and public keys: the target delegatee's public key is $X_0 = A = g^a$. For $i = 1, \ldots, n$, other public keys are set as $X_i = g^{x_i}$ with $x_i \xleftarrow{R} \mathbb{Z}_p^*$.

Queries: $\mathcal{A}_2$'s hash and signing queries are handled exactly as in the proof of limited proxy security. Namely, $\mathcal{B}_2$ fails if $\mathcal{A}_2$ asks for a signature on a message m for which $H(m) = B^\mu$ and responds consistently otherwise.

When $\mathcal{A}_2$ outputs her forgery $\sigma^{\star(L)} = (\sigma_0^\star, \ldots, \sigma_{2L-2}^\star)$ at level L, $\mathcal{B}_2$ is successful if $H(m^\star) = B^{\mu^\star}$, for some $\mu^\star \in \mathbb{Z}_p^*$, and extracts an admissible $(2L-1)$-uple as done in the proof of limited proxy security.

Delegator security This security property is proven under the mCDH assumption. Given an adversary $\mathcal{A}_3$ with advantage ε, we outline an algorithm $\mathcal{B}_3$ that has probability $O(\varepsilon/q_s)$ of finding g^{ab} given $(g, A = g^a, A' = g^{1/a}, B = g^b)$.

Public key generation: as previously, the target public key is defined as $X_0 = A = g^a$. Remaining public keys are set as $X_i = g^{x_i}$ for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$ for $i = 1, \ldots, n$. This time, $\mathcal{A}_3$ aims at producing a first level forgery and is granted *all* re-signature keys, including R_{0j} and R_{j0}. For indexes (i, j) s.t. $i, j \neq 0$, $\mathcal{B}_3$ sets $R_{ij} = g^{x_i/x_j}$. If $i = 0$, it calculates $R_{0j} = A^{1/x_j} = g^{a/x_j}$. If $j = 0$ (and thus $i \neq 0$), $\mathcal{B}_3$ computes $R_{i0} = A'^{x_i} = g^{x_i/a}$ to $\mathcal{A}_3$.

$\mathcal{A}_3$'s queries are dealt with exactly as for previous adversaries. Eventually, $\mathcal{A}_3$ produces a first level forgery $\sigma^{\star(1)}$ for a new message $m^\star$. Then, $\mathcal{B}_3$ can extract g^{ab} if $H(m) = (g^b)^{\mu^\star}$ for some $\mu^\star \in \mathbb{Z}_p^*$, which occurs with probability $O(1/q_s)$ using Coron's technique [14]. Otherwise, $\mathcal{B}_3$ fails.

External security We finally show that an external security adversary $\mathcal{A}_4$ also allows breaking the $(L-1)$-FlexDH assumption almost exactly as in the proof of limited proxy security. The simulator $\mathcal{B}_4$ is given an instance $(g, A = g^a, B = g^b)$. As previously, $\mathcal{B}_4$ must "program" the random oracle H hoping that its output will be $H(m^\star) = B^{\mu^\star}$ (where $\mu^\star \in \mathbb{Z}_p^*$ is known) for the message $m^\star$ that the forgery $\sigma^{\star(L)}$ pertains to. The difficulty is that $\mathcal{B}_4$ must also be able to answer signing queries made on $m^\star$ for all but one signers. Therefore, $\mathcal{B}_4$ must guess which signer $i^\star$ will be $\mathcal{A}_4$'s prey beforehand. At the outset of the game, it thus chooses an index $i^\star \xleftarrow{R} \{1, \ldots, N\}$. Signer $i^\star$'s public key is set as $X_{i^\star} = A = g^a$. All other signers $i \neq i^\star$ are assigned public keys $X_i = g^{x_i}$ for which $\mathcal{B}_4$ knows the matching secret x_i and can thus always answer signing queries.

Hash queries and signing queries involving $i^\star$ are handled as in the proof of limited proxy security. When faced with a re-signing query from i to j for a valid signature $\sigma^{(\ell)}$ at level $\ell \in \{1, \ldots, L\}$, $\mathcal{B}_4$ ignores $\sigma^{(\ell)}$ and simulates a first level signature for signer j. The resulting signature $\sigma'^{(1)}$ is then turned into a $(\ell+1)^{\text{th}}$-level signature and given back to $\mathcal{A}_4$. A re-signing query thus triggers a signing query that only causes failure if $H(m)$ differs from g^μ for a known

$\mu \in \mathbb{Z}_p^*$.

When $\mathcal{A}_4$ forges a signature at level L, $\mathcal{B}_4$ successfully extracts a $(2L-1)$-Flexible Diffie-Hellman tuple (as $\mathcal{B}_1$ and $\mathcal{B}_2$ did) if $H(m^\star) = (g^b)^{\mu^\star}$ and if it correctly guessed the identity $i^\star$ of the target signer. If $\mathcal{A}_4$'s advantage is ε, we find $O(\varepsilon/(N(q_s + q_{rs} + 1)))$ as a lower bound on $\mathcal{B}_4$'s probability of success, q_s and q_{rs} being the number of signature and re-signature queries respectively. □

5 A Scheme in the Standard Model

Several extensions of BLS signatures have a standard model counterpart when Waters' technique supersedes random oracle manipulations (e.g. [24]). Likewise, we can very simply twist our method and achieve the first unidirectional PRS scheme (even including single hop ones) that avoids resorting to the random oracle model.

The scheme is, *mutatis mutandis*, quite similar to our first construction. Standard model security thus comes at the expense of a trusted setup to generate system parameters.

5.1 The Single Hop Variant

As in [31], n denotes the length of messages to be signed. Arbitrary long messages can be signed if we first apply a collision-resistant hash function with n-bit outputs, in which case n is part of the security parameter.

The scheme requires a trusted party to generate common public parameters. However, this party can remain off-line after the setup phase.

Global-setup(λ, n)**:** given security parameters λ, n, this algorithm chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^\lambda$, generators $g, h \overset{R}{\leftarrow} \mathbb{G}$ and a random $(n+1)$-vector $\overline{u} = (u', u_1, \ldots, u_n) \overset{R}{\leftarrow} \mathbb{G}^{n+1}$. The latter defines a function $F : \{0,1\}^n \rightarrow \mathbb{G}$ mapping n-bit strings $\mathsf{m} = m_1 \ldots m_n$ (where $m_i \in \{0,1\}$ for all $i \in \{0,1\}$) onto $F(\mathsf{m}) = u' \cdot \prod_{i=1}^n u_i^{m_i}$. The public parameters are

$$\mathsf{cp} := \{\mathbb{G}, \mathbb{G}_T, g, h, \overline{u}\}.$$

Keygen(λ)**:** user i sets his public key as $X_i = g^{x_i}$ for a random $x_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$.

ReKeygen(x_j, X_i)**:** given user j's private key x_j and user i's public key X_i, generate the unidirectional re-signature key $R_{ij} = X_i^{1/x_j} = g^{x_i/x_j}$ that will be used to translate signature from i into signatures from j.

Sign$(1, \mathsf{m}, x_i)$**:** to sign a message $\mathsf{m} = m_1 \ldots m_n \in \{0,1\}^n$ at level 1, pick $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$ at random and compute

$$\sigma^{(1)} = (\sigma_0, \sigma_1) = (h^{x_i} \cdot F(\mathsf{m})^r, g^r)$$

Sign$(2, \mathsf{m}, x_i)$**:** to generate a second level signature on $\mathsf{m} = m_1 \ldots m_n \in \{0,1\}^n$, choose $r, t \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and compute

$$\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2, \sigma_3) = (h^{tx_i} \cdot F(\mathsf{m})^r, g^r, X_i^t, g^t)$$

Re-Sign$(1, \mathsf{m}, \sigma^{(1)}, R_{ij}, X_i, X_j)$**:** on input of a message $\mathsf{m} \in \{0,1\}^n$, the re-signature key $R_{ij} = g^{x_i/x_j}$, a signature $\sigma^{(1)} = (\sigma_0, \sigma_1)$ and public keys X_i, X_j, check the validity of $\sigma^{(1)}$ w.r.t signer i by testing if

$$e(\sigma_0, g) = e(X_i, h) \cdot e(F(\mathsf{m}), \sigma_1) \tag{3}$$

If $\sigma^{(1)}$ is a valid, it can be turned into a signature on behalf of j by choosing $r', t \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computing

$$\begin{aligned}\sigma^{(2)} &= (\sigma_0', \sigma_1', \sigma_2', \sigma_3') = (\sigma_0^t \cdot F(\mathsf{m})^{r'}, \sigma_1^t \cdot g^{r'}, X_i^t, R_{ij}^t) \\ &= (h^{tx_i} \cdot F(\mathsf{m})^{r''}, g^{r''}, X_i^t, g^{tx_i/x_j})\end{aligned}$$

where $r'' = tr + r'$. If we set $\tilde{t} = tx_i/x_j$, we have

$$\sigma^{(2)} = (\sigma_0', \sigma_1', \sigma_2', \sigma_3') = (h^{\tilde{t}x_j} \cdot F(\mathsf{m})^{r''}, g^{r''}, X_j^{\tilde{t}}, g^{\tilde{t}}).$$

Verify$(1, \mathsf{m}, \sigma^{(1)}, X_i)$**:** the validity of a 1st level signature $\sigma^{(1)} = (\sigma_1, \sigma_2)$ is checked by testing if (3) holds.

Verify$(2, \mathsf{m}, \sigma^{(2)}, X_i)$**:** a signature $\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2, \sigma_3)$ at level 2 is accepted for the public key X_i if the following conditions are true.

$$\begin{aligned}e(\sigma_0, g) &= e(\sigma_2, h) \cdot e(F(\mathsf{m}), \sigma_1') \\ e(\sigma_2, g) &= e(X_i, \sigma_3).\end{aligned}$$

To the best of our knowledge, the above scheme is the first unidirectional PRS in the standard model and solves another problem left open in [12] where all constructions require the random oracle model. Like the scheme of section 4, it can be scaled into a multi-hop system.

5.2 The Multi-Hop Extension

At levels $\ell \geq 2$, algorithms Sign, Re-Sign and Verify are generalized as follows.

Sign$(\ell+1, m, x_i)$**:** to sign $\mathsf{m} \in \{0,1\}^n$ at level $\ell+1$, user i picks $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, $(t_1, \ldots, t_\ell) \stackrel{R}{\leftarrow} (\mathbb{Z}_p^*)^\ell$ and outputs $\sigma^{(\ell+1)} = (\sigma_0, \ldots, \sigma_{2\ell+1}) \in \mathbb{G}^{2\ell+2}$ where

$$\begin{cases}\sigma_0 = h^{x_i t_1 \cdots t_\ell} \cdot F(\mathsf{m})^r \\ \sigma_1 = g^r \\ \sigma_k = g^{x_i t_1 \cdots t_{\ell+2-k}} & \text{for } k \in \{2, \ldots, \ell+1\} \\ \sigma_k = g^{t_{k-\ell-1}} & \text{for } k \in \{\ell+2, \ldots, 2\ell+1\}.\end{cases}$$

Re-Sign$(\ell+1, \mathsf{m}, \sigma^{(\ell+1)}, R_{ij}, X_i, X_j)$**:** on input of a message $\mathsf{m} \in \{0,1\}^*$, the re-signature key $R_{ij} = g^{x_i/x_j}$, a purported $(\ell+1)^{\text{th}}$-level signature

$$\begin{aligned}\sigma^{(\ell+1)} &= (\sigma_0, \ldots, \sigma_{2\ell+1}) \\ &= (h^{x_i t_1 \cdots t_\ell} \cdot F(\mathsf{m})^r, g^r, g^{x_i t_1 \cdots t_\ell}, g^{x_i t_1 \cdots t_{\ell-1}}, \ldots, g^{x_i t_1}, g^{t_1}, \cdots, g^{t_\ell}) \in \mathbb{G}^{2\ell+2}\end{aligned}$$

and public keys X_i, X_j, check the correctness of $\sigma^{(\ell+1)}$ under X_i. If valid, $\sigma^{(\ell+1)}$ is translated for X_j by sampling $r' \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, $(r_0, r_1, \ldots, r_\ell) \stackrel{R}{\leftarrow} (\mathbb{Z}_p^*)^{\ell+1}$ and setting $\sigma^{(\ell+2)} = (\sigma_0', \ldots, \sigma_{2\ell+3}') \in \mathbb{G}^{2\ell+4}$ where

$$\begin{cases}\sigma_0' = \sigma_0^{r_0 \cdots r_\ell} \cdot F(\mathsf{m})^{r'} \\ \sigma_1' = \sigma_1^{r_0 \cdots r_\ell} \cdot g^{r'} \\ \sigma_k' = \sigma_k^{r_0 \cdots r_{\ell+2-k}} & \text{for } k \in \{2, \ldots, \ell+1\} \\ \sigma_{\ell+2}' = X_i^{r_0} \\ \sigma_{\ell+3}' = R_{ij}^{r_0} \\ \sigma_k' = \sigma_{k-2}^{r_{k-\ell-3}} & \text{for } k \in \{\ell+4, \ldots, 2\ell+3\}.\end{cases}$$

If we define $\tilde{t}_0 = r_0 x_i/x_j$, $r'' = r_0 \cdots r_\ell + r'$ and $\tilde{t}_k = r_k t_k$ for $k = 1, \ldots, \ell$, we observe that

$$\sigma^{(\ell+2)} = (h^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_\ell} \cdot F(\mathsf{m})^{r''}, g^{r''}, g^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_\ell}, g^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_{\ell-1}}, \ldots, g^{x_j \tilde{t}_0}, g^{\tilde{t}_0}, \ldots, g^{\tilde{t}_\ell}).$$

Verify$(\ell+1, \mathsf{m}, \sigma^{(\ell+1)}, X_i)$**:** a candidate signature $\sigma^{(\ell+1)} = (\sigma_0, \ldots, \sigma_{2\ell+1})$ is verified by testing if the following equalities hold:

$$\begin{aligned} e(\sigma_0, g) &= e(h, \sigma_3) \cdot e(F(\mathsf{m}), \sigma_1) \\ e(\sigma_k, g) &= e(\sigma_{k+1}, \sigma_{2\ell+3-k}) \text{ for } k \in \{2, \ldots, \ell\} \\ e(\sigma_{\ell+1}, g) &= e(X_i, \sigma_{\ell+2}). \end{aligned}$$

5.3 Security

Theorem 2. *The scheme with L levels (and thus at most $L-1$ hops) is a secure unidirectional PRS under the $(L-1)$-FlexDH and mCDH assumptions.*

Proof. The proof is very similar to the one of theorem 1 and is detailed in the full version of the paper [23]. □

6 Single-Hop Schemes in the Chosen Key Model

This section shows a simple way to modify the single-hop versions of our schemes so as to prove their security in the plain public key model and dispense with the knowledge of secret key assumption. We outline the required modifications in our first scheme but they can be applied to our standard model system as well.

The idea is to randomize the generation of re-signature keys, the shape of which becomes reminiscent of Waters signatures. Using techniques that were initially proposed for identity-based encryption [10], we can then prove security results without positioning ourselves in the KOSK model.

Global-setup(λ)**:** is as in section 4.
Keygen(λ)**:** user i's public key is $pk_i = (X_i = g^{x_i}, Y_i = g^{y_i})$ for random $x_i, y_i \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$.
ReKeygen(x_j, y_j, pk_i)**:** given x_j, y_j and $pk_i = (X_i, Y_i)$, this algorithm outputs the re-signature key

$$R_{ij} = (R_{ij1}, R_{ij2}) = (X_i^{1/x_j} \cdot Y_j^r,\ X_j^r)$$

for a random $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and where $(X_j, Y_j) = (g^{x_j}, g^{y_j})$.
Sign$(1, x_i, m)$**:** outputs $\sigma^{(1)} = H(m)^{x_i} \in \mathbb{G}$ as in section 4.
Sign$(2, x_i, m)$**:** to sign $m \in \{0,1\}^*$ at level 2, user i chooses $s, t \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computes

$$\begin{aligned} \sigma^{(2)} &= (\sigma_0, \sigma_1, \sigma_2, \sigma_3) \\ &= (H(m)^{x_i t},\ X_i^t,\ g^t \cdot Y_i^s,\ X_i^s). \end{aligned}$$

Re-Sign$(1, m, \sigma^{(1)}, R_{ij}, pk_i, pk_j)$**:** given the re-signature key $R_{ij} = (R_{ij1}, R_{ij2})$, a signature $\sigma^{(1)} \in \mathbb{G}$ and public keys $pk_i = (X_i, Y_i)$, $pk_j = (X_j, Y_j)$, check the validity of $\sigma^{(1)}$ w.r.t signer i by testing $e(\sigma^{(1)}, g) = e(H(m), X_i)$. If valid, $\sigma^{(1)}$ is turned into a signature on behalf of j by choosing $s', t \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computing

$$\begin{aligned} \sigma^{(2)} &= (\sigma_0', \sigma_1', \sigma_2', \sigma_3') \\ &= ({\sigma^{(1)}}^t,\ X_i^t,\ R_{ij1}^t \cdot Y_j^{s'},\ R_{ij2}^t \cdot X_j^{s'}) \\ &= (H(m)^{x_i t},\ X_i^t,\ g^{t x_i / x_j} \cdot Y_j^{rt+s'},\ X_j^{rt+s'}) \end{aligned}$$

If we set $\tilde{t} = t x_i / x_j$ and $\tilde{s} = rt + s'$, we have

$$\sigma^{(2)} = (H(m)^{x_j \tilde{t}},\ X_j^{\tilde{t}},\ g^{\tilde{t}} \cdot Y_j^{\tilde{s}},\ X_j^{\tilde{s}}).$$

Verify$(1, m, \sigma^{(1)}, pk_i)$**:** accept $\sigma^{(1)}$ if $e(\sigma^{(1)}, g) = e(H(m), X_i)$.
Verify$(2, m, \sigma^{(2)}, pk_i)$**:** accept $\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2, \sigma_3)$ w.r.t. $pk_i = (X_i, Y_i)$ if the following relations hold.

$$e(\sigma_0, g) = e(\sigma_1, H(m)) \quad e(\sigma_2, X_i) = e(g, \sigma_1) \cdot e(Y_i, \sigma_3)$$

The above scheme features a comparable efficiency to the one of section 4 with signatures that are only slightly longer at level 2. We were unfortunately unable to turn it into a multi-hop system.

From a security standpoint, we also need fewer assumptions in the proofs since the 1-Flexible Diffie-Hellman assumption suffices.

Theorem 3. *The single-hop scheme is secure in the chosen-key model under the* 1*-FlexDH assumption.*

Proof. We can prove the result without resorting to the modified CDH assumption and using only the 1-Flexible Diffie-Hellman problem. Let $(g, A = g^a, B = g^b)$ be a given instance of the latter.

External security and limited proxy security For these notions, the proofs work out almost exactly as in the proof of theorem 1. The only difference is in the generation of users' public keys $pk_i = (X_i, Y_i)$: the first component X_i is chosen as in the proof of theorem 1 whilst Y_i is set as $Y_i = X_i^{y_i}$ for randomly drawn exponents $y_i \xleftarrow{R} \mathbb{Z}_p^*$. When the adversary eventually outputs a forgery

$$(\sigma_0^\star, \sigma_1^\star, \sigma_2^\star, \sigma_3^\star) = (H(m^\star)^{axt},\ X^t,\ g^t \cdot Y^r,\ X^r),$$

w.r.t. to a honest user's public key $(X = A^x, Y = X^y)$ (where $x, y \in \mathbb{Z}_p^*$ are random exponents initially chosen by the simulator), one can compute $(\sigma_0^\star, \sigma_1^\star, \sigma_2^\star / \sigma_3^{\star y})$ and use it as a forgery against our scheme of section 4. Namely, if $H(m^\star) = B^{\mu^\star}$ and $X = A^x$ for known values $x, \mu^\star \in \mathbb{Z}_p^*$, the simulator obtains a triple

$$(C^{ab}, C^a, C) = \Big(\sigma_0^\star,\ \sigma_1^{\star\mu^\star},\ (\frac{\sigma_2^\star}{\sigma_3^{\star y}})^{x\mu^\star}\Big),$$

which solves the problem instance.

Delegatee security The proof is as in theorem 1 but the adversary is given a single honest user's public key.

Delegator security From an adversary $\mathcal{A}$ with advantage ε and making q_s signing queries, we build an algorithm $\mathcal{B}$ that finds g^{ab} with probability $O(\varepsilon/q_s)$.

System parameters: $\mathcal{A}$ is provided with public parameters $\{\mathbb{G}, \mathbb{G}_T, g, \mathcal{O}_H\}$ where $\mathcal{O}_H$ is the random oracle.
Key generation: the delegator's public key is defined as $pk_0 = (X_0, Y_0) = (A, g^{y_0})$ for a random $y_0 \xleftarrow{R} \mathbb{Z}_p^*$.
Oracle queries:

- $\mathcal{A}$'s random oracle queries and signing queries are handled using Coron's technique [14] as in the proof of theorem 1 (we have thus again a degradation factor of $O(q_s)$ in the reduction).

- *Delegation queries*: at any time $\mathcal{A}$ can supply a public key $pk = (X, Y)$ (without having to reveal the underlying secret) and ask oracle $\mathcal{O}_{dlg}(.)$ to generate a re-signature key on behalf of the delegator 0 using pk as the delegatee's public key. Since we have $(X_0 = A, Y_0 = g^{y_0})$ for a known exponent y_0, $\mathcal{B}$ picks $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and returns
$$(R_1, R_2) = (g^{ry_0},\ X_0^r \cdot X^{-1/y_0}). \tag{4}$$
If we define $\tilde{r} = r - x/(ay_0)$, where $x = \log_g(X)$, we see that (R_1, R_2) has the correct shape since
$$X^{1/a} \cdot Y_0^{\tilde{r}} = X^{1/a} \cdot Y_0^r \cdot (g^{y_0})^{-\frac{x}{ay_0}} = g^{ry_0}$$
and $X_0^{\tilde{r}} = X_0^r \cdot A^{-\frac{x}{ay_0}} = X_0^r \cdot X^{-1/y_0}$. We observe that $\mathcal{B}$ can compute both parts of (4) without knowing $x = \log_g(X)$ or $y = \log_g(Y)$.

After a number of queries, $\mathcal{A}$ comes up with a first level forgery that allows computing g^{ab} as in the proof of theorem 1. Unlike what happens in the latter, $\mathcal{B}$ does not need $g^{1/a}$ at any time during the simulation and we only need the 1-Flexible Diffie-Hellman assumption. □

7 Can one achieve constant-size multi-hop signatures?

While highly desirable, unidirectional multi-hop PRS with constant-size signatures turn out to be very hard to construct. We give arguments explaining why they seem out of reach with the current state of knowledge.

Trivially, if the Re-Sign algorithm increases the size of signatures (even by a single bit), then we inevitably end up with a linear size in the number of delegations. Intuitively, multi-hop unidirectional systems therefore provide either constant or linear sizes. It seems very unlikely that one will be able to come up with logarithmic-size signatures for instance. This apparently indicates that, regardless of how many times signatures get translated, they should remain in the same signature space (which sounds hardly compatible with the pursued unidirectionality). Nonetheless, not all unidirectional schemes do lengthen signatures upon translation: if implemented with appropriate parameters, the first proposal of [12] features the same signature size at both levels (though signatures have different shapes). However, it does not lend itself to a multi-use extension: to translate a signature, the proxy uses a piece of it as an exponent to exponentiate the re-signature key, which hampers length-preserving re-iterations of the process.

Up to now, all known unidirectional proxy re-cryptography primitives make use of bilinear maps. Unfortunately, those tools still fall short of reaching the aforementioned purpose. Pairing-based schemes often let proxies replace a component of the original ciphertext or signature by its pairing with the proxy key. Multiple hops are impossible if we leave the resulting pairing value inside the re-signature since no bilinear map is defined over the target group $\mathbb{G}_T$. To circumvent this issue, our approach postpones the computation of the pairing until the verification by blinding its arguments and introducing them into transformed signatures. Unfortunately, this inevitably increases their length at each conversion.

We are not claiming that constant signature sizes are impossible to obtain. But it turns out that new ideas and techniques should be developed to reach this goal.

8 Generic hardness of ℓ-FlexDH in bilinear groups

To provide more confidence in the ℓ-FlexDH assumption we give a lower bound on the computational complexity of the ℓ-FlexDH problem for generic groups equipped with bilinear maps.

In [22], Kunz-Jacques and Pointcheval define a family of computational problems that enables to study variants of the CDH problem in the generic group model. Let $\mathcal{A}$ be an adversary in this model and $\varphi(X_1, \dots, X_k, Y_1, \dots, Y_\ell)$ be a multivariate polynomial whose coefficients might depend on $\mathcal{A}$'s behavior. For values of $x_1, \dots, x_k$ chosen by the simulator, and knowing their encodings, the goal of $\mathcal{A}$ is to compute the encodings of $y_1, \dots, y_\ell$ such that

$$\varphi(x_1, \dots, x_k, y_1, \dots, y_\ell) = 0.$$

All elements manipulated by $\mathcal{A}$ are linear polynomials in $x_1, \dots, x_k$ and some new random elements introduced via the group oracle. Let us denote P_i the polynomial corresponding to y_i (it is a random variable), Kunz-Jacques and Pointcheval proved the following result.

Theorem 4 ([22]). *Let $d = \deg(\varphi)$ and $\mathsf{P_m}$ be an upper bound for the probability*

$$\Pr[\varphi(X_1, \dots, X_k, P_1(X_1, \dots, X_k), \dots, P_\ell(X_1, \dots, X_k)) = 0]$$

Then the probability that $\mathcal{A}$ wins after q_G queries satisfies

$$\mathsf{Succ}(q_G) \leq \mathsf{P_m} + \frac{(3q_G + k + 2)}{2p} + \frac{d}{p}.$$

The choice $\phi(X_1, X_2, Y_1, \dots, Y_{\ell+1}) = Y_{\ell+1} - X_1 X_2 Y_1 \dots Y_\ell$ implies the generic hardness of the ℓ-FlexDH problem. It is almost straightforward to prove that the Kunz-Jacques-Pointcheval result also holds in generic bilinear groups where the ℓ-FlexDH problem thus remains intractable. The details are given in the full version of the paper.

Theorem 5. *Let $d = \deg(\varphi)$ and P_m be an upper bound for the probability*

$$\Pr[\varphi(X_1, \dots, X_k, P_1(X_1, \dots, X_k), \dots, P_\ell(X_1, \dots, X_k)) = 0]$$

Then the probability that $\mathcal{A}$ wins after q_G oracle queries to the group operations in $\mathbb{G}$, $\mathbb{G}_T$ to the bilinear map e satisfies

$$\mathsf{Succ}(q_G) \leq P_m + \frac{(3q_G + k + 2)}{p} + \frac{d}{p}.$$

9 Conclusions and Open Problems

We described the first multi-use unidirectional proxy re-signatures, which solves a problem left open at CCS 2005. Our random-oracle-based proposal also offers efficiency improvements over existing solutions at the first level. The other scheme additionally happens to be the first unidirectional PRS in the standard model. We finally showed how to construct single-hop schemes in the chosen-key model.

Two major open problems remain. First, it would be interesting to see if multi-level unidirectional PRS have efficient realizations under more classical intractability assumptions. A perhaps more challenging task would be to find out implementations – if they exist at all – of such primitives where the size of signatures and the verification cost do not grow linearly with the number of translations.

Acknowledgements

The authors thank Mark Manulis and the anonymous referees for their comments. The first author acknowledges the support of the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.). The second author is supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT and by the French *Agence Nationale de la Recherche* through the PACE project.

References

1. M. Abe and S. Fehr, *Perfect NIZK with Adaptive Soundness.*, in Vadhan [30], pp. 118–136.
2. J. H. An, Y. Dodis, and T. Rabin, *On the Security of Joint Signature and Encryption.*, Advances in Cryptology - Eurocrypt 2002 (L. R. Knudsen, ed.), Lect. Notes Comput. Sci., vol. 2332, Springer, 2002, pp. 83–107.
3. G. Ateniese, K. Fu, M. Green, and S. Hohenberger, *Improved proxy re-encryption schemes with applications to secure distributed storage.*, ACM Trans. Inf. Syst. Secur. **9** (2006), no. 1, 1–30.
4. M. Bellare and G. Neven, *Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma.*, (2006), 390–399.
5. M. Bellare and A. Palacio, *The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols.*, Advances in Cryptology - Crypto 2004 (M. K. Franklin, ed.), Lect. Notes Comput. Sci., vol. 3152, Springer, 2004, pp. 273–289.
6. ———, *Towards Plaintext-Aware Public-Key Encryption Without Random Oracles.*, Advances in Cryptology - Asiacrypt 2004 (P. J. Lee, ed.), Lect. Notes Comput. Sci., vol. 3329, Springer, 2004, pp. 48–62.
7. M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.*, Proceedings of the First ACM Conference on Computer and Communications Security (D. Denning, R. Pyle, R. Ganesan, R. Sandhu, and V. Ashby, eds.), ACM Press, 1993, pp. 62–73.
8. M. Blaze, G. Bleumer, and M. Strauss, *Divertible Protocols and Atomic Proxy Cryptography.*, Advances in Cryptology - Eurocrypt'98 (K. Nyberg, ed.), Lect. Notes Comput. Sci., vol. 1403, Springer, 1998, pp. 127–144.
9. A. Boldyreva, *Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme.*, 6th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2003 (Y. Desmedt, ed.), Lect. Notes Comput. Sci., vol. 2567, Springer, 2003, pp. 31–46.
10. D. Boneh and X. Boyen, *Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles.*, Advances in Cryptology - Eurocrypt 2004 (C. Cachin and J. Camenisch, eds.), Lect. Notes Comput. Sci., vol. 3027, Springer, 2004, pp. 223–238.
11. D. Boneh, B. Lynn, and H. Shacham, *Short Signatures from the Weil Pairing.*, J. Cryptology **17** (2004), no. 4, 297–319.
12. X. Boyen, Q. Mei, and B. Waters, *Proxy Re-Signatures: New Definitions, Algorithms, and Applications.*, Proceedings of the 12th ACM Conference on Computer and Communications Security (V. Atluri, B. Pfitzmann, and P. McDaniel, eds.), ACM Press, 2005, 310-319.
13. R. Canetti and S. Hohenberger, *Chosen-Ciphertext Secure Proxy Re-Encryption.*, Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007 (P. Ning, S. De Capitani di Vimercati, and P. F. Syverson, eds.), ACM Press, 2007, pp. 185–194.
14. J.-S. Coron, *On the Exact Security of Full Domain Hash.*, Advances in Cryptology - Crypto 2000 (M. Bellare, ed.), Lect. Notes Comput. Sci., vol. 1880, Springer, 2000, pp. 229–235.
15. I. B. Damgård, *Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks.*, Advances in Cryptology - Crypto'91 (J. Feigenbaum, ed.), Lect. Notes Comput. Sci., vol. 576, Springer, 1992, pp. 445–456.
16. A. W. Dent, *The Hardness of the DHK Problem in the Generic Group Model.*, Tech. Report 2006/156, IACR eprint, 2006.
17. R. Granger and N. P. Smart, *On Computing Products of Pairings.*, Tech. Report 2006/172, IACR eprint, 2006.
18. M. Green and G. Ateniese, *Identity-Based Proxy Re-encryption.*, Applied Cryptography and Network Security, ACNS 2007 (J. Katz and M. Yung, eds.), Lect. Notes Comput. Sci., vol. 4521, Springer, 2007, pp. 288–306.
19. S. Hohenberger, *Advances in Signatures, Encryption, and E-Cash from Bilinear Groups.*, Ph.D. thesis, MIT, mai 2006.
20. S. Hohenberger, G. N. Rothblum, a. shelat, and V. Vaikuntanathan, *Securely Obfuscating Re-encryption.*, in Vadhan [30], pp. 233–252.
21. A.-A. Ivan and Y. Dodis, *Proxy Cryptography Revisited.* , Proceedings of the Network and Distributed System Security Symposium, NDSS 2003 (V. Gligor and M. Reiter, eds.), The Internet Society, 2000, pp. 143–154.

22. S. Kunz-Jacques and D. Pointcheval, *About the Security of MTI/C0 and MQV.*, Fifth Conference on Security and Cryptography for Networks, SCN'06 (R. De Prisco and M. Yung, eds.), Lect. Notes Comput. Sci., vol. 4116, Springer, 2006, pp. 156–172.
23. B. Libert and D. Vergnaud, *Multi-Use Unidirectional Proxy Re-Signatures.*, Tech. Report 0802.1113, Computing Research Repository, 2008.
24. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters, *Sequential Aggregate Signatures and Multisignatures Without Random Oracles.*, Advances in Cryptology - Eurocrypt 2006 (S. Vaudenay, ed.), Lect. Notes Comput. Sci., vol. 4004, Springer, 2006, pp. 465–485.
25. M. Mambo, K. Usuda, and E. Okamoto, *Proxy Signatures for Delegating Signing Operation.*, Proceedings of the Third ACM Conference on Computer and Communications Security (L. Gong and J. Stern, eds.), ACM Press, 1996, pp. 48–57.
26. M. Naor, *On Cryptographic Assumptions and Challenges.*, Advances in Cryptology - Crypto 2003 (D. Boneh, ed.), Lect. Notes Comput. Sci., vol. 2729, Springer, 2003, pp. 96–109.
27. T. Ristenpart and S. Yilek, *The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks*, Advances in Cryptology - Eurocrypt 2007 (M. Naor, ed.), Lect. Notes Comput. Sci., vol. 4515, Springer, 2007, pp. 228–245.
28. C. P. Schnorr, *Efficient signature generation by smart cards.*, J. Cryptology **4** (1991), no. 3, 161–174.
29. J. Shao, Z. Cao, L. Wang, and X. Liang, *Proxy Re-signature Schemes Without Random Oracles.*, Progress in Cryptology - Indocrypt 2007 (K. Srinathan, C. Pandu Rangan, and Moti Yung, eds.), Lect. Notes Comput. Sci., vol. 4859, Springer, 2007, pp. 197–209.
30. S. P. Vadhan (ed.), *Theory of cryptography, 4th theory of cryptography conference, tcc 2007, amsterdam, the netherlands, february 21-24, 2007, proceedings*, Lect. Notes Comput. Sci., vol. 4392, Springer, 2007.
31. B. Waters, *Efficient Identity-Based Encryption Without Random Oracles.*, Advances in Cryptology - Eurocrypt 2005 (R. Cramer, ed.), Lect. Notes Comput. Sci., vol. 3494, Springer, 2005, pp. 114–127.