A MULTI-USE UNI-DIRECTIONAL PROXY RE-SIGNATURE SCHEME

and a new security definition for uni-directional proxy re-signature schemes

T. Jonas Özgan

DIPLOMARBEIT

im Studiengang Informatik
der

Angefertigt am:

B-IT: Bonn-Aachen International Center for Information Technology

CoSec: Computer Security Group

Prof. Dr. Joachim von zur Gathen
Priv.-Doz. Dr. Adrian Spalka

Erklärung

Ich, T. Jonas Özgan, Matrikel-Nr. 1523134, versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Bonn, den 18. April 2011

Acknowledgements

First i would like to thank Prof. Dr. von zur Gathen for giving me the opportunity to write this thesis under his supervision and also Dr. Spalka for his assistance in evaluating this thesis.

Most importantly i would like to thank Dr. Nüsken for his never ending *patience* and his brilliant ideas which helped me to accomplish this thesis.

I also would like to thank my family, my parents and my friends for their never ending support and backup which kept me alive and going.

To my masters...

Contents

1. Introduction

1.1. Motivation. Digital signatures are mathematical schemes demonstrating the authenticity of a document. A valid signature suggests that the authenticated document belongs to a known signer, and it also assures the integrity of the document. Although in practice there are a variety of types of digital signatures, generally we can say that, a digital signature can be used to prove the originality of a document while also providing a mechanism to protect the signed document against alteration. In the age of information and with the increasing availability of high speed internet, users come in contact with a large variety of digital media such as music, books, articles, photos, and so on. Also the popularity of smart phones, portable multi-media players and next generation gaming consoles has turned the digital world into a big trading and sharing place, spread all over the world. The demand for digital media has made the trading of digital content a very lucrative business. Sensing this opportunity many companies started to offer digital media, some of the well known examples are Apple's iTunes, Microsoft Windows Media and Napster. These companies use content protection systems to protect their files against illegal distribution but also to authenticate them. Since different companies use different content protection mechanisms the users cannot use the digital content on devices of their choice. This lack of interoperability gives reasons for the illegal usage and distribution of digital media and also slows down the growth of the industry. To achieve interoperable content protection mechanisms, but also for e-cash and e-passport systems, a special type of digital signature is required. As in the real world, when a user sells his CD to another user, the banderol on the CD proves its originality to the new owner. In this selling process which could take place for example on a flea market, the users do not need any interaction with the record company or with the bank that issued money. As in this simple trading example we require a digital signature that can be given away easily. This means that, as in the real world, the users who want to exchange their files with a currency should only need to interact with each other and not with the content providers or the currency owner. A good approach to achieve this kind of digital trading is the concept of *proxy re-cryptography* specially *proxy re-signatures*.

1.2. Concept. In this thesis we will analyze in detail the proposal of Libert & Vergnaud (2008a), a *multi-use unidirectional proxy re-signature*. We choose to analyze this signature scheme because of its translation property and its possible usages in practice such as digital content protection systems and electronic cash systems. Our aim was to introduce the signature scheme step by

step for a comprehensive understanding of its structure and relations to be able to analyze its security and its efficiency.

1.3. Contributions. In this work we provide a new security definition for uni-directional proxy re-signatures. The shortcomings of the original security definition from Ateniese & Hohenberger (2005) such as the artificial splitting of the security definition and the unnatural limitation of the adversaries, motivated us to construct a new game based security definition. In Shao *et al.* (2010) the authors also point out the shortcomings and the unnecessary complexity of the old security definition and provide another security definition for uni-directional proxy re-signatures with certain probabilities. Similar to our proposal the authors consider a generic adversary with access to as much information as possible to overcome the shortcomings of the old security definition. Differing from their proposal, we provide a simple graph algorithm to keep track of the adversary's queries and to detect trivial forgeries. As we show in Part III, simple modifications to the graph algorithm seem to make our definition also valid for different types of proxy re-signatures. Therefore, we believe that our new security definition provides the necessary flexibility to be adapted and used for different types of proxy re-signatures with different properties.

Further we can list the following:

- We explained the idea behind the construction of the signature by extending a short signature (Boneh *et al.* 2004) into a multi-use proxy re-signature step by step. We used the additive notation for the signature scheme instead of the multiplicative one. We changed the numbering of the indices of the signature elements and as well as the used coefficients to provide a more intuitive understanding of the signature scheme. We showed how the verification equations are related to the elements of the signature. This allowed us to develop a graphical notation to demonstrate the relation between signature elements. We also expressed the construction of the signature scheme by decomposing it into simple building blocks.

- We tried to analyze the efficiency of the signature scheme from two different angles: (1) The amount of randomness and (2) the length of the signature. Unfortunately there was no hint or discussion in the original publication we could make use of. To be able to analyze the length of the signature we introduced a new problem called the *chain shortening problem*. We provided some insight what would it mean to have a shorter

signature or even if this was possible. We also analyzed the amount of randomness used by the signing and re-signing algorithms and pointed out the implications of using lesser or related coefficients.

- We put together the possible usages of proxy re-signatures which were suggested in different publications. We focused our attention on Taban *et al.* (2006) to point out the practical importance of proxy re-signatures in content protection systems.

1.4. Related Work. The concept of *proxy re-cryptography* was first introduced in Blaze, Bleumer & Strauss (1998) as *atomic proxy cryptography*, in which a semi trusted proxy can convert signatures of `Aylin` into the signatures of `Boris` on the same message. However, in this process the proxy can not sign arbitrary messages on behalf of both parties `Aylin` and `Boris`. This cryptographic primitive received renewed interest with the publication Ateniese & Hohenberger (2005) in which the authors provided useful security definitions and introduced two new *proxy re-signature schemes*, (1) multi-use bidirectional and (2) single-use unidirectional. The security of both of these schemes was proven in the *random oracle model* (Bellare & Rogaway 1993). The authors left open the challenge to find a *multi-use unidirectional* scheme which was also secure in the *standard model*. In Libert & Vergnaud (2008a) the authors proposed the first *multi-use unidirectional proxy re-signature* scheme which is also secure in the *standard model*, after a slight modification. This scheme was based on bilinear maps, unlike the later proposal of a *multi-use unidirectional* scheme in Sunitha & Amberker (2009) which is based on factoring. Note that a *proxy re-signature* is not the same as a *proxy signature*. In the *proxy re-signature* scheme a proxy "translates" a valid and publicly verifiable signature $\sigma_A(m)$ of `Aylin` on a message m into $\sigma_B(m)$ one from `Boris` on the same message. However, *proxy signatures* allow `Aylin` to delegate her signing rights to `Boris` but only if `Proxy` cooperates. The general idea is to divide `Aylin`'s secret into two shares. `Boris` and `Proxy` only receive one share each so they can jointly generate signatures on behalf of `Aylin` on the same message. Clearly *proxy signatures* have a completely different application area than *proxy re-signatures*.

1.5. Structure of the Thesis. In the following sections we will introduce the signature scheme step by step with its theoretical background and design idea. This thesis is divided into five main parts.

- In Part I, we start with the foundations of elliptic curve based cryptography. Remembering the definition of an elliptic curve, we show that

the points on an elliptic curve with the point of infinity form an abelian group. After defining pairings and how to calculate them, we discuss some elliptic curve based digital signatures.

- In Part II, we start with the discussion of possible methods of transferring a signature of user `Aylin` to user `Boris`. Since the trivial methods show fatal deficits we formulate our requirements to a transferable signature. We then build up step by step the signature scheme from Libert & Vergnaud (2008a) by transferring a short signature (Boneh *et al.* 2004) once and generalize this idea into a multi-use scheme. In this process we also see the relations of the signature elements in a graphical form which enables us to define the signature with building blocks. We finish this chapter with formally writing down the signature scheme.

- Part III begins with the introduction to the cryptographic assumptions underlying the signature scheme. We continue with discussion of the adversary model and the two environments in which the adversary is simulated. We then introduce our new security definition for uni-directional proxy re-signatures. This allows us to compare our new security definition to the original one after recalling the security definition from Ateniese & Hohenberger (2005) and outlining its limitations. We then prove that the signature scheme is secure for the new security definition in the random oracle model. After modifying the signature scheme slightly we also prove the security of the signature scheme in the standard model.

- In Part IV we analyze the efficiency of the signature from two angles. First, we introduce a new problem class called the *chain shortening problem*, which helps to understand the length of the signature. Second, we analyze the amount of used randomness to build the signature and discuss the results of using lesser or related coefficients.

- In Part V we put together the possible applications of *proxy re-signatures* especially focusing on the interoperable digital rights management proposal from Taban *et al.* (2006).

- Finally in Part VI we outline and conclude the results achieved in this thesis.

Part I
Elliptic curve cryptography

Elliptic curves have a rich history and have been studied by mathematicians over a century before they have become popular in cryptographic research and applications in the last thirty years. In 1985, Neal Koblitz and Victor Miller independently proposed to use elliptic curves for public-key cryptographic systems. However, the acceptance of elliptic curve based crypto-systems came in the late nineties when accredited standard organizations such as the American National Standards Institute (ANSI) specified protocols based on elliptic curves. At present there are numerous applications and publications on elliptic curves. This shows that there has been an extensive amount of research carried out in this area. The aim of this chapter is give an introduction to elliptic curves and pairing based elliptic curve cryptography.

2. Elliptic Curves

There are many different ways of introducing elliptic curves such as starting with the *canon ball problem* as Washington (2008) or starting more algebraic like Werner (2002). We start right away with the definition:

DEFINITION 2.1. *An elliptic curve E over a field $\mathbb{F}$ is defined by an equation in the form*

$$E: \ y^2 = x^3 + ax + b \tag{1.1}$$

where $a, b \in \mathbb{F}$ and $\Delta \neq 0$. Here, the discriminant $\Delta = -16(4a^3 + 27b^2)$ of the curve is used to exclude singular cases.

This equation is called the *simplified Weierstrass equation* or just the *Weierstrass equation*. Note here that usually in literature the elliptic curves are introduced by what is known as the *generalized Weierstrass equation*. However one can show that the *generalized Weierstrass* form of an elliptic curve can always be transformed into the *simplified Weierstrass* equation above if the characteristic of $\mathbb{F}$, is neither 2 nor 3. The *generalized Weierstrass equation* as well as the simplification process are explained in detail in Washington (2008) and Werner (2002). For any extension $\mathbb{K}$ of the field $\mathbb{F}$, $\mathbb{K} \supseteq \mathbb{F}$ we can consider the set of $\mathbb{K}$-rational points

$$E(\mathbb{K}) \ := \{\mathbf{O}\} \cup \{(x, y) \in \mathbb{K} \times \mathbb{K} |\ y^2 = x^3 + ax + b\}. \tag{2.2}$$

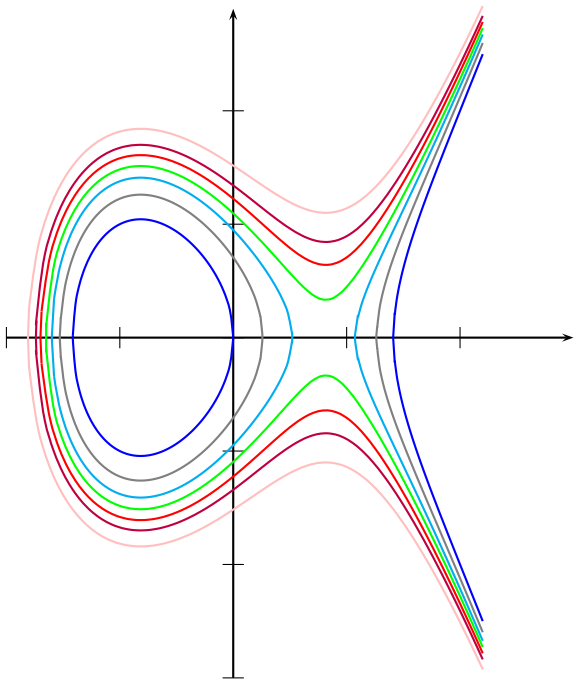


Figure 2.1: Elliptic Curves $E : y^2 = x^3 - 2x + \alpha$

Figure 2.1 shows different elliptic curves given by $y^2 = x^3 - 2x + \alpha$.

The point **O** is called the point at infinity which, allows us to prove the group structure of points on the curve. One can imagine this point sitting somewhere up high on top of the y-axis. Visually, consider a two dimensional plane (like a sheet of paper) on which an elliptic curve is drawn. Starting from a point on the curve, an ant could walk in two directions on the curve. In each direction the ant would fall off the plane and come to some "undefined" place. The algebraic nature of these places (points) is all the same which is **O**, the point at infinity. Understanding the algebraic nature of this point requires an introduction to projective space which can be found in Werner (2002).

3. The group law

We can actually introduce a group structure on an elliptic curve. This, in turn,is used to construct elliptic curve based cryptosystems.

DEFINITION 3.1. *Let E be an elliptic curve over a finite field $\mathbb{F}$ given in the*

Weierstrass form, and $P_1 = (x_1, y_1)$ *and* $P_2 = (x_2, y_2)$ *two different points on* E*. To add* P_1 *and* P_2 *we draw a line passing through* P_1 *and* P_2*. This line intersects* E *in a third point which we call* $P_3 = (x_3, y_3)$*. Then* P_3 *is reflected along the* x*-axis by changing the sign of the* y*-coordinate and this is* $P_1 + P_2$*.*

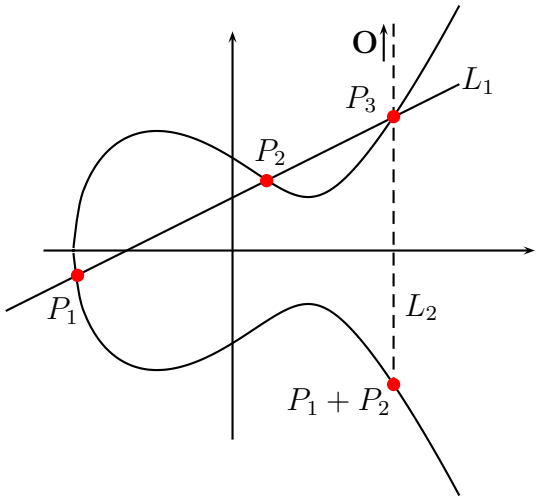


Figure 3.1: $P_1 + P_2$ on an elliptic curve

Let us first assume that $P_1 \neq P_2$ and none of them is $\mathbf{O}$. The slope of L_1 is

$$m = \frac{y_2 - y_1}{x_2 - x_1}. \tag{3.2}$$

In case where $x_1 = x_2$, the line L_1 will be vertical and for now we assume that this is not the case. The equation of L_1 is

$$y = m(x - x_1) + y_1.$$

Now substituting this in the Weierstrass equation of E we obtain

$$m(x - x_1) + y_1)^2 = x^3 + ax + b.$$

The resulting equation will be in the form

$$0 = x^3 - m^2x^2 + \cdots$$

This cubic polynomial has three roots and we know two of them, namely x_1 and x_2. Since $(x_1 + x_2 + x_3) = m^2$ we obtain

$$x_3 = m^2 - x_1 - x_2 \text{ and } y_3 = m(x_3 - x_1) + y_1.$$

Reflecting this across the x-axis yields $P_1 + P_2 = P_4 = (x_4, y_4)$ with

$$x_4 = m^2 - x_1 - x_2 \text{ and } y_4 = m(x_1 - x_3) - y_1.$$

Now consider the case where $P_1 = P_2 = (x_1, y_1)$. This means that the line L_1 is tangent to E at P_1. Since $P_1 = P_2$ we use implicit differentiation to find out the slope m of L_1

$$\frac{dy}{dx} = m = \frac{3x_1^2 + a}{2y_1}. \tag{3.3}$$

Again two roots, or better one double root, of the cubic polynomial

$$0 = x^3 - m^2x^2 + ...$$

are known and we can find out the third root. The same technique as above gives us for $P_1 + P_1 = P_4 = (x_4, y_4)$ the values

$$x_4 = m^2 - 2x_1,\ y_4 = m(x_1 - x_4) - y_1.$$

this is called *point doubling* (Figure 3.2).

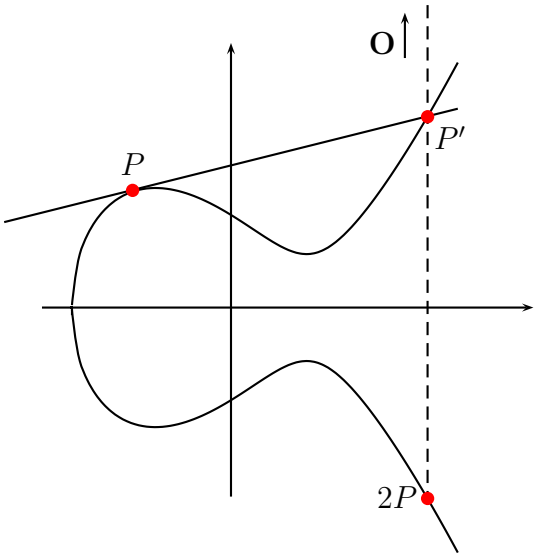


Figure 3.2: Point Doubling

Now consider two points P_1 and P_2 where $x_1 = x_2$ and $y_1 \neq y_2$. The line through P_1 and P_2 is parallel to the y-axis thus the third point of interception

is **O**. Reflecting **O** across the x-axis is again **O** since the algebraic nature of "all" **O**'s are the same as mentioned above. Therefore here we get $P_1 + P_2 = \mathbf{O}$.

Finally, assume that one of the points is **O**. Similar to last the case above, the line through P_1 and **O** is vertical. The third point of interception is the reflection of P_1 across the x-axis, reflecting it again will result back in P_1. Thus $P_1 + \mathbf{O} = P_1$. Here we can see that with this definition of an addition over E, the point **O** is behaving as a *neutral element*.

Now we summarize the addition cases from above to define an addition on an elliptic curve.

DEFINITION 3.4. *Let E be an elliptic curve in Weierstrass form: $y^2 = x^3 + ax + b$. Given two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ on E, P_1, $P_2 \neq \mathbf{O}$ and m is the slope of the line through P_1 and P_2 (see equations (3.2) and (3.3)). Define an addition by $P_1 + P_2 := P_4 = (x_4, y_4)$ where:*

(i) $x_4 = m^2 - x_1 - x_4$ *and* $y_4 = m(x_1 - x_4) - y_1$, *if* $x_1 \neq x_2$.

(ii) $P_4 = \mathbf{O}$ *if* $x_1 = x_2$ *but* $y_1 \neq y_2$, *ie.* P_1 *and* P_2 *are symmetric with respect to the* x*-axis.*

(iii) $x_4 = m^2 - 2x_1$ *and* $y_4 = m(x_1 - x_4) - y_1$, *if* $P_1 = P_2$ *and* $y \neq 0$, *ie. the point* P_1 *is a double.*

(iv) $P_4 = \mathbf{O}$ *if* $P_1 = P_2$ *and* $y = 0$, *ie. the point* P_1 *is a double root on the* x*-axis.*

The definition Definition 3.4 allows us to formulate the following theorem.

THEOREM 3.5. *The points on an elliptic curve E form an abelian group with the addition operation as defined above with* **O** *as the neutral element. In particular,*

(i) $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$ *for all* P_1, P_2, P_3 *on* E *(the group is associative),*

(ii) $P + \mathbf{O} = P$ *for all* P *on* E (**O** *is the group's neutral element),*

(iii) for any P *on* E *there is a* P' *on* E *which satisfies* $P + P' = \mathbf{O}$ *(existence of inverses),*

(iv) $P_1 + P_2 = P_2 + P_1$ *for all* P_1, P_2 *on* E *(the group is commutative).*

We already pointed out above how **O** behaves as a neutral element. For a detailed proof of the group properties see Washington (2008) or Werner (2002).

4. Pairings

In this section we want to make a proper introduction to pairings which will be sufficient for understanding the later sections. A pairing is a function mapping a pair of points from two groups $\mathbf{G}_1$ and $\mathbf{G}_2$ to another group $\mathbf{G}_T$. In many applications all three groups are usually of prime order n. This mapping, often noted as $e(\cdot,\cdot)$, has some properties which are especially attractive in cryptographic settings. We study the applications of pairings in cryptographic settings in the next section.

DEFINITION 4.1 (Torsion points). *Consider an elliptic curve E defined over $\mathbb{F}_q$ and an integer n not divisible by the characteristic of $\mathbb{F}_q$. The set $E[n]$ of n-torsion points is given by*

$$E[n] = \{P \in E(\overline{\mathbb{F}}_q) | nP = \mathbb{O}\},$$

where $\overline{\mathbb{F}}_q$ is the algebraic closure of $\mathbb{F}_q$. In other words the set of n-torsion points consists of all points $P \in E(\overline{\mathbb{F}}_q)$ which have order dividing n.

Now we introduce a bilinear pairing in a basic setting.

DEFINITION 4.2 (Pairing). *Let $E(\mathbb{F}_q)$ be an elliptic curve defined over $\mathbb{F}_q$, $\mathbf{G}_1$ and $\mathbf{G}_2$, $\mathbf{G}_T$ three groups usually of prime order n. Typically $\mathbf{G}_1$ and $\mathbf{G}_2$ are subgroups of $E[n]$ and $\mathbf{G}_T$ is a subgroup of $\mathbb{F}_{q^k}^{\times}$, where k is called the embedding degree if n is the minimal integer dividing $q^k - 1$. Then there exists a map:*

$$e\colon \mathbf{G}_1 \times \mathbf{G}_2 \longrightarrow \mathbf{G}_T,$$

which satisfies the following conditions.

(i) The map e is bilinear:

- $e(P_1 + Q_1, P_2) = e(P_1, P_2)e(Q_1, P_2)$,
- $e(P_1, P_2 + Q_2) = e(P_1, P_2)e(P_1, Q_2)$,

for all $P_1, Q_1 \in \mathbf{G}_1$ and $P_2, Q_2 \in \mathbf{G}_2$.

(ii) The map e is non-degenerate:

- $e(P_1, P_2) \neq 1$,

for some $P_1 \in \mathbf{G}_1$ and $P_2 \in \mathbf{G}_2$.

(iii) For practical reasons we require that e is efficiently computable with respect to the input size which is $\Theta(\log q)$ bits since $P, Q \in E(\mathbb{F}_q)$.

An example of this is the ***Weil pairing*** which has the form

$$e_n \colon E[n] \times E[n] \longrightarrow \mu_n \tag{4.3}$$

for an elliptic curve $E(\mathbb{F}_q)$ where

$$\mu_n = \{x \in \overline{\mathbb{F}}_q | x^n = 1\}$$

denotes the set of n^{th} roots of unity in $\overline{\mathbb{F}}_q$. The *Weil pairing* satisfies the following conditions

1. e_n is bilinear:
   - $e_n(P + Q, R) = e_n(P, R)e_n(Q, R)$,
   - $e_n(P, Q + R) = e_n(P, Q)e_n(P, R)$

   for all $P, Q, R \in E[n]$.
2. e_n is non-degenerate: If $e_n(P, Q) = 1$ for all $Q \in E[n]$, then $P = \mathbf{O}$.

Notice that differing from Definition 4.2 we use here only one additive group $E[n]$ instead of two, which is called the *symmetric case*. For the definition of the *Weil pairing* and a proof of the listed properties above see Washington (2008).

Based on the *Weil pairing* it is possible to construct other pairings, an example of this is the ***modified Tate-Lichtenbaum pairing*** which has the form

$$\tau_n \colon E(\mathbb{F}_q)[n] \times E(\mathbb{F}_q)/nE(\mathbb{F}_q) \longrightarrow \mu_n \subseteq \mathbb{F}_q^{\times}, \tag{4.4}$$

for an elliptic curve $E(\mathbb{F}_q)$ with $n \mid q-1$. Here $E(\mathbb{F}_q)[n]$ denotes the elements of E with coordinates in $\mathbb{F}_q$ and of order dividing n, where μ_n is again the group of the n^{th} roots of unity as defined above.

The *modified Tate-Lichtenbaum pairing* can be constructed from the *Weil pairing* by

$$\tau_n(P, Q) = e_n(P, R - \phi_q R),$$

where $P \in E(\mathbb{F}_q)[n], Q \in E(\mathbb{F}_q), R \in E(\overline{\mathbb{F}}_q)$ and $nR = Q$. Here

$$\phi_q \colon \begin{array}{ccc} \overline{\mathbb{F}}_q & \longrightarrow & \overline{\mathbb{F}}_q, \\ x & \longmapsto & x^q \end{array}$$

is called the q-th power of the *Frobenius endomorphism.*

In contrast to (4.4) the original ***Tate-Lichtenbaum pairing*** has the following form:

$$\langle \cdot, \cdot \rangle_n \colon E(\mathbb{F}_q)[n] \times E(\mathbb{F}_q)/nE(\mathbb{F}_q) \longrightarrow \mathbb{F}_q^\times/(\mathbb{F}_q^\times)^n, \tag{4.5}$$

Note that since we obtain here a coset in $\mathbb{F}_q^\times$ mod n-th powers by taking the n-th root of μ_n, the *modified Tate-Lichtenbaum* pairing is more suitable for practical applications than the original one and these pairings can be calculated quickly (Washington 2008).

We mentioned above that for practical applications we require that the pairing is efficiently computable. However, we require a basic understanding of divisor theory before discussing the computation of pairings. The following is mostly taken from Joux (2002); Meffert (2009); Nüsken (2010) and Washington (2008).

4.1. Divisors. In this section we will make a brief introduction into divisor theory. Roughly speaking a divisor D is an element of the group generated by the points of the curve E. It is used to keep track of poles and zeros. The function f is said to have a zero at point P if it takes the value 0 at P, similarly it has a pole at P if it takes the value $\mathbf{O}$ at P. Then D can be written as a finite sum $D := \sum_i a_i(P_i)$ where each P_i is a point on E and each a_i is an integer. Given a function f from the set of rational maps in the coordinates of points x, y we build a divisor $\mathrm{div}(f)$ from the zeros and poles of f by forming the formal sum of zeros and poles with their multiplicity.

Remember Section 3 where we introduced the addition of points on elliptic curves. We used a line passing through two points P_1 and P_2 on the curve E and concluded that this line has to intersect E at a third point P_3 (Figure 3.1). We actually used the solutions to the function $0 = mx - y + c$ to calculate the coordinates of the point P_3. Now consider a non-trivial function $f = ax+by+c$ and assume that it passes through three points P_1, P_2 and P_3. If $b \neq 0$ then $P_1, P_2, P_3 \neq \mathbf{O}$ and f has a triple pole in $\mathbf{O}$. Thus

$$\mathrm{div}(ax + by + c) = [P_1] + [P_2] + [P_3] - 3[\mathbf{O}].$$

On the other hand if $b = 0$ then the line passes through P_3, $-P_3$ and $\mathbf{O}$. We obtain

$$\mathrm{div}(x - x_3) = [P_3] + [-P_3] - 2[\mathbf{O}].$$

In Figure 3.1 this is L_2 connecting P_3 and $P_1 + P_2$. Now if we rewrite $-P_3 =$

$P_1 + P_2$ we get

$$\text{div}\left(\frac{ax + by + c}{x - x_3}\right) = [P_1] + [P_2] - [P_1 + P_2] - [\mathbf{O}], \tag{4.6}$$

where x_3 is the x-coordinate of P_3. Equivalently, we have

$$[P_1 + P_2] + [\mathbf{O}] + \text{div}(\frac{ax + by + c}{x - x_3}) = [P_1] + [P_2].$$

Since we can always draw a line through two given points $P_1, P_2 \in E$ we can replace a divisor $[P_1] + [P_2]$ with $[P_1 + P_2] + [\mathbf{O}]$ plus the divisor of some other function.

We observe that the sum of points of a divisor is $\mathbf{O}$ and the degree of the divisor is 0. However proving this requires more theory than introduced here, for a *proper* introduction to divisor theory we refer to Washington (2008).

4.2. Calculation of pairings. In this section we will introduce the algorithm from Miller (1986) for the computation of pairings. We start with the definition of the Tate-Lichtenbaum pairing and continue with its calculation.

DEFINITION 4.7 (Tate-Lichtenbaum pairing). *Let $E(\mathbb{F}_q)$ be an elliptic curve and fix a prime n which is not divisible by the characteristic of $\mathbb{F}_q$. Further, let k be the smallest integer such that $n \mid q^k - 1$ (embedding degree). Also assume that f_P is a function with divisor $n[P + R] - n[R]$ for some R, and $Q_1 - Q_2 = Q$ such that $P + R$, R, Q_1, Q_2 are all different and non-zero. We define the* Tate-Lichtenbaum pairing *by*

$$\langle \cdot, \cdot \rangle_n \colon \begin{array}{rcl} E(\mathbb{F}_q)[n] \times E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k}) & \longrightarrow & \mathbb{F}_{q^k}^{\times}/(\mathbb{F}_{q^k}^{\times})^n, \\ (P, Q) & \longmapsto & \langle P, Q \rangle_n = \frac{f_P(Q_1)}{f_P(Q_2)} \end{array} .$$

and the modified Tate-Lichtenbaum pairing *by*

$$\tau_n \colon \begin{array}{rcl} E(\mathbb{F}_q)[n] \times E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k}) & \longrightarrow & \mu_n \subseteq \mathbb{F}_{q^k}^{\times}, \\ (P, Q) & \longmapsto & \langle P, Q \rangle_n^{\frac{q^k - 1}{n}} \end{array} .$$

Here $E(\mathbb{F}_q)[n]$ denotes the elements of E with coordinates in $\mathbb{F}_q$ and of order dividing n and μ_n the set of n^{th} roots of unity as defined above. The group $E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k})$ is the set of equivalence classes of points of $E(\mathbb{F}_q)$ where two points are considered equivalent if their difference is another point of order n. The group $\mathbb{F}_{q^k}^{\times}/(\mathbb{F}_{q^k}^{\times})^n$ isomorphic to μ_n is the set of equivalence classes of the elements of $\mathbb{F}_q$ where two elements considered to be equivalent if they are the same up to the multiplication with an n^{th} power.

Since the final exponentiation with $\frac{q^k-1}{n}$ can be handled, the main goal is actually to find that function f_P and calculate the pairing

$$\langle P, Q\rangle_n = \frac{f_P(Q_1)}{f_P(Q_2)},$$

such that the divisor of f_P and the divisor $D_Q := [Q_1] - [Q_2]$ are disjoint.

Now we want to make a connection between n and the function f_P which will allow us to calculate this pairing. We first define a function f_i for $i \geq 0$ such that

$$\text{(i)} \qquad \operatorname{div}(f_i) = D_i := i[P+R] - i[R] - [iP] + [\mathbf{O}]$$

with $P, Q, R \in E$ as above. We observe here that

$$\operatorname{div}(f_n) = n[P+R] - n[R] - \underbrace{[nP]}_{[\mathbf{O}]} + [\mathbf{O}] = \operatorname{div}(f_P)$$

since P is a torsion point and thus $nP = \mathbf{O}$. This means if we can compute the value

$$\frac{f_n(Q_1)}{f_n(Q_2)} = \frac{f_P(Q_1)}{f_P(Q_2)} = \langle P, Q\rangle_n \tag{4.8}$$

we have reached our goal.

We note that for $i = 0$ we get

$$\frac{f_0(Q_1)}{f_0(Q_2)} = 1$$

since

$$D_0 := 0[P+R] - 0[R] - [0P] + [\mathbf{O}] = 0.$$

For $i = 1$ we have

$$D_1 := [P+R] - [R] - [P] + [\mathbf{O}].$$

Now assume that $\ell = ax + by + c$ is the line through P and R, $v = x + d$ the vertical line through $P + R$ and $\mathbf{O}$. Then we obtain

$$\frac{f_1(Q_1)}{f_1(Q_2)} = \frac{\frac{ax+by+c}{x+d}\,|_{(x,y)=Q_1}}{\frac{ax+by+c}{x+d}\,|_{(x,y)=Q_2}}.$$

Now we want to use f_0 and f_1 to calculate the values $f_2, f_3, \ldots, f_n$ such that we can compute the desired value (4.8). Therefore, assuming that the values

$$\frac{f_j(Q_1)}{f_j(Q_2)} \quad \text{and} \quad \frac{f_k(Q_1)}{f_k(Q_2)}$$

are already calculated for some integers j, k, we want to derive a solution for

$$\frac{f_{j+k}(Q_1)}{f_{j+k}(Q_2)}.$$

As defined in (i) above f_j and f_k have the divisors

$$\text{div}(f_j) = D_j := j[P+R] - j[R] - [jP] + [\mathbf{O}], \tag{1}$$

$$\text{div}(f_k) = D_k := k[P+R] - k[R] - [kP] + [\mathbf{O}]. \tag{2}$$

Let $\ell = ax + by + c$ the line through jP and kP, and let $v = x + d$ the vertical line through $(j+k)P$. Recalling equation (4.6) we get

$$\text{div}\left(\frac{ax+by+c}{x+d}\right) = [jP] + [kP] - [(j+k)P] - [\mathbf{O}]. \tag{3}$$

Adding (1),(2) and (3) we get

$$\text{div}\left(f_j f_k \frac{ax+by+c}{x+d}\right) = D_{j+k} := (j+k)[P+R] - (j+k)[R] - [(j+k)P] + [\mathbf{O}].$$

Consequently, we obtain

$$\frac{f_{j+k}(Q_1)}{f_{j+k}(Q_2)} = \frac{f_j(Q_1)}{f_j(Q_2)} \cdot \frac{f_k(Q_1)}{f_k(Q_2)} \cdot \frac{\frac{ax+by+c}{x+d}\,|_{(x,y)=Q_1}}{\frac{ax+by+c}{x+d}\,|_{(x,y)=Q_2}}$$

as the evaluation of f_{j+k} at Q_1 and Q_2 which is the required value. This means that to calculate the value of f_{j+k} at Q_1 and Q_2 we we only need the values of f_j and f_k there and the points jP and kP.

The following algorithm from Miller (1986) starts with f_1 and successively uses point doubling and adding to reach up to f_n.

Miller's Algorithm.

Input: Points $P, R, Q_1, Q_2 \in E$ and the final index n where $nP = \mathbf{O}$.

Output: The value of $\langle P, Q\rangle_n = \frac{f_P(Q_1)}{f_P(Q_2)}$ where $\text{div}(f_P) = n[P+R] - n[R] - [nP] + [\mathbf{O}]$.

1. Compute $\ell = ax + by + c$ the line through P and R.
2. Compute $v = x + d$ the vertical line through $P + R$ and $\mathbf{O}$.
3. Compute $f_1 \leftarrow \frac{\frac{ax+by+c}{x+d}|_{(x,y)=Q_1}}{\frac{ax+by+c}{x+d}|_{(x,y)=Q_2}}$.
4. Let $f \leftarrow f_1$, $J \leftarrow P$.
5. Write $n = (n_{r-1}, \ldots, n_1 n_0)$ in base 2.
6. For $i = r-2, \ldots, 0$ do 7–17
7. Let $\ell = ax + by + c$ be the tangent at J.
8. $S \leftarrow 2J$.
9. Let $v = x + d$ be the vertical line through S.
10. $f \leftarrow f^2 \cdot \frac{\ell}{v}\,|_{Q_1} \cdot \frac{v}{\ell}\,|_{Q_2}$.
11. $J \leftarrow S$.
12. If $n_i = 1$ then
13. Let $\ell = ax + by + c$ be the line through J and P.
14. $S \leftarrow J + P$.
15. Let $v = x + d$ be the vertical line through S.
16. $f \leftarrow f \cdot f_1 \cdot \frac{\ell}{v}\,|_{Q_1} \cdot \frac{v}{\ell}\,|_{Q_2}$.
17. $J \leftarrow S$.
18. Return f.

The runtime of Miller's Algorithm is determined by the point adding and doubling steps which depend on the group order n. Recall that the input size is $\Theta(\log q)$ then the algorithm makes $O(k \log q)$ point operations which is considered to be too slow for practical purposes. This results from the choice of q^k which is $q^k \approx 2^{1024}$ bits (or even $q^k \approx 2^{2048}$) and is related to the security of the signature scheme. We will discuss the choice of q^k in the context of the security of the signature scheme later in Part III.

Miller's Algorithm is used as a basis for the calculation of pairings and there have been numerous proposals to speed up and optimize it for specific groups and different pairings. For example, the recent publication Costello & Stebila (2010) proposes a precomputation based modification of Miller's Algorithm that is 37% faster than the original one and $19,5\%$ faster than other precomputation based approaches. For a detailed discussion we refer to some of the publications in this area such as Costello & Stebila (2010); Galbraith, Harrison & Soldera (2002); Ian Blake & Xu (2004)

5. Elliptic curve based cryptosystems

Having introduced elliptic curves and bilinear pairings we now look at some examples of elliptic curve based cryptosystems. Since we later aim to analyze

a signature scheme in the rest of this thesis, we will concentrate on signature schemes based on elliptic curves and pairings.

5.1. El-Gamal type signature scheme. In El-Gamal (1985) the El-Gamal signature scheme was introduced. It is based on the hardness of the *discrete logarithm problem* in certain groups. We explain here an elliptic curve variant of the El-Gamal Signature Scheme.

Given an elliptic curve E over a finite field $\mathbb{F}_q$ (where the *discrete log problem* is hard) and a point $P \in E(\mathbb{F}_q)$ with a large (prime) order n. Let also $H\colon \{0,1\}^* \to \mathbb{Z}_n$ be a hash function and $f\colon E(\mathbb{F}_q) \to \mathbb{Z}_n$ be a function mapping the points on E to integers. `Boris` wants to verify a signature $\sigma_m(s, R)$ which was signed by `Aylin` on a message m. He first retrieves `Aylin`'s public key $A = aP \in E(\mathbb{F}_q)$ (where a is the private key). After that his verifying algorithm does:

1. Compute $v_1 \leftarrow f(R)A + sR$.
2. Compute $v_2 \leftarrow H(m)P$.
3. Check whether $v_1 \stackrel{?}{=} v_2$.

We deduce the signing process from the verifying equation. Consider that $R = rP$ where r is a random element with $\gcd(r, n) = 1$ and we also know that for a valid signature we must have $v_1 = v_2$. Thus

$$\begin{aligned} v_1 &= f(R)A + sR \\ &= f(R)aP + srP \\ &= (f(R)a + sr)P \\ &= (f(R)a + sr)P \stackrel{?}{=} v_2 = H(m)P. \end{aligned}$$

This means that $f(R)a+sr = H(m)$ which directly gives us the signing equation as

$$s = r^{-1}(H(m) - af(R)).$$

Altogether we have:

EC El-Gamal sign. `Aylin` signs.
Public input: The group $E(\mathbb{F}_q)$, the base point P and its order n, the functions $H\colon \{0,1\}^* \to \mathbb{Z}_n$ and $f\colon E(\mathbb{F}_q) \to \mathbb{Z}_n$.
Input: `Aylin`'s private key $a \in \mathbb{Z}_n^\times$, the message $m \in \{0,1\}^*$.
Output: Signature $\sigma_m(s, R)$.

1. Choose a random $r \xleftarrow{\$} \mathbb{Z}_n^\times$ with $\gcd(r, n) = 1$.
2. Compute $R \leftarrow rP$.
3. Compute $s \leftarrow r^{-1}\left(H(m) - af(R)\right) \mod n$.
4. Return $\sigma_m \leftarrow (s, R)$.

EC El-Gamal verify. `Boris` verifies.

Public input: The group $E(\mathbb{F}_q)$, the base point P and its order n, the functions $H\colon \{0,1\}^* \to \mathbb{Z}_n$ and $f\colon E(\mathbb{F}_q) \to \mathbb{Z}_n$ and the public key A of `Aylin`.
Input: The message $m \in \{0,1\}^*$ and a signature $\sigma_m(s, R)$.
Output: {ACCEPT, REJECT}.

1. Compute $v_1 \leftarrow f(R)A + sR$.
2. Compute $v_2 \leftarrow H(m)P$.
3. If $v_1 = v_2$ then
4. Return ACCEPT.
5. Else
6. Return REJECT.

For security reasons we require that the function f which converts points into integers in the field $\mathbb{Z}_n$ allows preimage computation. A simple example of this function would be just to take the x-coordinate of a given point, ie $f((x, y)) = x$. This would result in at most two points yielding the same output under f which is acceptable.
An attacker `Charly` can forge the signature if he can calculate the discrete logarithm a from $A = aP$ or by finding a collision in the hash function such as $H(m) = H(m')$. Both of them are assumed to be hard problems. However `Aylin` needs to be careful when signing messages. Assume that `Charly` obtains two signatures (m, R, s) and (m', R, s') signed with the same r. Then the two equations for s and s' are

$$rs \equiv H(m) - af(R),$$
$$rs' \equiv H(m') - af(R).$$

Subtracting these would give `Charly` $r(s - s') \equiv H(m) - H(m') \mod n$. `Charly` can now compute r and with that he can obtain the private key a of `Aylin`.

5.2. Elliptic curve digital signature algorithm (ECDSA). The El-Gamal signature scheme in the raw form above is rarely used in practice. The National Institute of Standards and Technology (NIST) proposed a variant of this scheme in 1991 which was called the *Digital Signature Algorithm* (U.S. Department of Commerce / National Institute of Standards and Technology 2000). A more recent version of it uses elliptic curves instead of multiplicative groups in finite fields. The algorithm is actually an El-Gamal type signature scheme with tiny modifications.

The main difference to the *El-Gamal* scheme above is in the verification procedure. Although the signature in the ECDSA scheme is computed exactly the same way as in the El-Gamal scheme, here a valid signature is verified by

$$\begin{aligned} rP &\stackrel{?}{=} s^{-1}H(m)P - s^{-1}f(R)A. \\ &= s^{-1}(H(m) - f(R)a)P \end{aligned}$$

Note that the *El-Gamal* system requires a total of three integer times point computations (which are expensive) in its verification equations $v_1 = f(R)A + sR$ and $v_2 = H(m)P$ where the *ECDSA* system only needs two in $s^{-1}H(m)P$ and $f(R)A$.

Note that there are again no special requirements to the function f. Also as above the signer has to be careful about signing different messages with the same random element r.

5.3. Short signatures. Boneh, Lynn & Shacham (2004) introduces *Short signatures from the Weil pairing*. The security of this signature scheme is based on the *computational Diffie-Hellman assumption* which is to find abP from a given triple (P, aP, bP), more in Part III. Compared to the ECDSA above, it has the same level of security but half of the length: 170 bits instead of 320. This is of course a significant improvement for low bandwidth systems and also for systems where humans are required to type in the signature.

Given an elliptic curve E over $\mathbb{F}_q$ and a point $P \in E(\mathbb{F}_q)$ generating the group $\mathbf{G}$. Further fix a hash function $H\colon \{0,1\}^* \to E(\mathbb{F}_q)$ that maps bit strings to points on the elliptic curve. Most importantly, choose a non-degenerate bilinear pairing $e\colon \mathbf{G} \times \mathbf{G} \to \mathbb{F}_q^\times$ satisfying Definition 4.2.

Suppose that `Boris` wants to verify `Aylin`'s signature σ on a message m. After retrieving her public key $A = aP$ his verifying algorithm does

1. Compute $u \leftarrow e(H(m), A)$.

2. Compute $v \leftarrow e(\sigma, P)$.

3. Check whether $v = u$.

Since we know that

$$u = e(H(m), A) = e(H(m), aP) = e(H(m), P)^a = e(aH(m), P),$$

we can satisfiy $v = u$ by assuming $aH(m) = \sigma$. Altogether we obtain

BLS SIGN. `Aylin` signs.

Public input: The Group $E(\mathbb{F}_q)$, the base point P and its order n, the hash function $H\colon \{0,1\}^* \to E(\mathbb{F}_q)$, the bilinear pairing $e\colon \mathbf{G} \times \mathbf{G} \to \mathbb{F}_q^\times$.
Input: `Aylin`'s private key $a \in \mathbb{Z}_q^\times$, the message $m \in \{0,1\}^*$.
Output: Signature σ_m.

1. Compute $\sigma_m \leftarrow aH(m)$.
2. Return σ_m.

BLS VERIFY. `Boris` verifies.

Public input: The Group $E(\mathbb{F}_q)$, the base point P and its order n, the function $H\colon \{0,1\}^* \to E(\mathbb{F}_q)$, the bilinear pairing $e\colon \mathbf{G} \times \mathbf{G} \to \mathbb{F}_q^\times$.
Input: `Aylin`'s public key A, the message $m \in \{0,1\}^*$, the signature σ_m.
Output: {ACCEPT, REJECT}.

1. Compute $u \leftarrow e(H(m), A)$.
2. Compute $v \leftarrow e(\sigma_m, P)$.
3. If $v = u$ then
4. Return ACCEPT.
5. Else
6. Return REJECT.

The security of this signature scheme is shown in the *random oracle model* introduced in Bellare & Rogaway (1993). Note that the construction of a hash function mapping to the points on the elliptic curve is not trivial. A detailed explanation of constructing such hash functions can be found in the original publication.

5.4. Multi-designated verifiers signature. In the mid 90's Jakobsson, Sako & Impagliazzo (1996) introduced the concept of *Designated verifier signatures* which was independently patented by Chaum (1996) as *private signatures*. In these proposals a signature could only be verified by a unique user chosen by the signer. The idea was that no one except the designated verifier could be convinced by a signature because the designated verifier could also produce the signature by himself. The authors suggested also an extension of their scheme to a set of designated verifiers. Later in Laguillaumie & Vergnaud (2007) this concept was formalized and *multi-designated verifiers signatures* were introduced. In their publication the authors introduced a *bi-designated verifiers signature* scheme which only can be validated by two designated verifiers (`Boris` and `Charly`) chosen by the signer `Aylin`. The idea behind the signature is that for a fourth party `David`, the signature states that either `Aylin` produced the signature or `Boris` and `Charly` together produced the signature.

Consider an elliptic curve E, two groups $\mathbf{G}$ and $\mathbf{H}$ of large prime order n and $P \in E$ a generator for $\mathbf{G}$. Chose a non-degenerate bilinear pairing $e\colon \mathbf{G} \times \mathbf{G} \to \mathbf{H}$ and a hash function $H\colon \{0,1\}^* \times \mathbf{H} \to \mathbf{G}$.

`Aylin` (the signer) chooses her private key as $a \xleftarrow{\$} \mathbb{Z}_n^\times$ where her public key $P_A = aP$. `Boris` (verifier #1) chooses his private key as $b \xleftarrow{\$} \mathbb{Z}_n^\times$ where his public key $P_B = bP$. `Charly` (verifier #2) chooses his private key as $c \xleftarrow{\$} \mathbb{Z}_n^\times$ where his public key $P_C = cP$.

As before suppose that `Boris` wants to verify a signature $\sigma_m = (Q_A, R, \ell)$ which was generated by `Aylin` for `Boris` and `Charly` the designated verifiers on a message m. Now `Boris` retrieves the public keys P_A and P_C . Then the verifying algorithm does

1. Compute $u \leftarrow e(P_A, P_C)^b$.
2. Compute $M \leftarrow H(m, u^\ell)$.
3. Compute $P_{BC} \leftarrow P_B + P_C$.
4. Check whether $e(Q_A, P_A)e(R, P_{BC}) = e(M, P)$.

If `Charly` instead of `Boris` verifies the signature he just replaces P_B with P_C and b with c. A signature σ_m is valid iff the equation $e(Q_A, P_A)e(R, P_{BC}) = e(M, P)$ holds. As before we construct $R = rP$ with a random integer r. Assuming that ℓ is another random integer, we can deduce the signing process

as

$$\begin{aligned} e(M,P) &= e(Q_A,P_A)e(R,P_{BC}) \\ &= e(Q_A,aP)e(rP,(b+c)P) \\ &= e(aQ_A,P)e(r(b+c)P,P) \\ &= e(aQ_A+r(b+c)P,P). \end{aligned}$$

This is satisfied if

$$\begin{aligned} M &= aQ_A + r(b+c)P, \text{ or} \\ Q_A &= a^{-1}(M - r(b+c)P). \end{aligned}$$

Now we summarize both processes.

MDVS SIGN. `Aylin` signs.

Public input: Two groups $\mathbf{G}$ and $\mathbf{H}$, the base point P and its order n, the non-degenerate bilinear pairing $e\colon \mathbf{G}\times\mathbf{G}\to\mathbf{H}$, the hash function $H\colon \{0,1\}^*\times\mathbf{H}\to\mathbf{G}$.

Input: `Aylin`'s private key $a\in\mathbb{Z}_n^\times$, the message $m\in\{0,1\}^*$, Public keys P_B and P_C of `Boris` and `Charly`.

Output: Signature σ_m.

1. Choose two random integers $r,\ell \xleftarrow{\text{R}} \mathbb{Z}_n^\times$.
2. Compute $P_{BC}\leftarrow P_B+P_C$.
3. Compute $u\leftarrow e(P_B,P_C)^a$.
4. Compute $M\leftarrow H(m,u^\ell)$.
5. Compute $Q_A\leftarrow a^{-1}(M-rP_{BC})$
6. Set $\sigma_m\leftarrow(Q_A,R,\ell)$.
7. Return σ_m.

MDVS VERIFY. `Charly` verifies.

Public input: Two groups $\mathbf{G}$ and $\mathbf{H}$, the base point P and its order n, the non-degenerate bilinear pairing $e\colon \mathbf{G}\times\mathbf{G}\to\mathbf{H}$, the hash function $H\colon \{0,1\}^*\times\mathbf{H}\to\mathbf{G}$.

Input: Public keys P_A and P_B of `Aylin` and `Boris`, the message $m\in\{0,1\}^*$, the signature σ_m.

Output: {ACCEPT, REJECT}.

1. Compute $u\leftarrow e(P_A,P_B)^c$.

2. Compute $M \leftarrow H(m, u^{\ell})$.
3. If $u = M$ then
4. Return ACCEPT.
5. Else
6. Return REJECT.

Taking into consideration the examples above one can still see the "touch" of *El-Gamal* in the equation $Q_A = a^{-1}(M - rP_{BC})$. This scheme is also secure in the *random oracle model*, as proven by the authors.

There have been numerous other publications about *designated verifiers signature schemes*. These suggest *universal designated verifiers signature* schemes and constructing designated verifiers signatures from *any* non-degenerate bilinear mapping for details see Laguillaumie & Vergnaud (2004); Saeednia *et al.* (2003); Steinfeld *et al.* (2003).

5.5. Proxy re-signatures. In Blaze, Bleumer & Strauss (1998) introduced a new cryptographic primitive called *atomic proxy cryptography*, in which a semi trusted proxy converts signatures of `Aylin` into signatures of `Boris` on the same message. However, in this process the proxy can not sign arbitrary messages for both parties. Until the publication of Ateniese & Hohenberger (2005), this cryptographic primitive was widely ignored by the cryptographic community. The authors revised the primitive and provided appropriate security definitions for the *random oracle model*. They also introduced two new proxy re-signature schemes (1) multi-use bidirectional and (2) single-use unidirectional.

5.5.1. Multi-use bidirectional scheme. Given the security parameter k fix two groups $\mathbf{G}_1$ and $\mathbf{G}_2$ of prime order n, a generator P for $\mathbf{G}_1$, a non-degenerate bilinear mapping $e\colon \mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$ and a hash function $H\colon \{0,1\}^* \to \mathbf{G}_1$.

In this type of schemes we have three parties to consider. `Aylin` the delegator with public key $A = aP$, the proxy who converts the signatures of `Aylin` into signatures of `Boris` identified by his public key $B = bP$. The proxy is able to do this with the *re-signature key* $R_{AB} = \frac{b}{a}$ which by assumption the proxy already has. Note that there are many *secure* ways of computing R_{AB}, an example is mentioned in Ateniese & Hohenberger (2005).

As before we will begin with the verification of a signature σ which was generated by `Aylin`. Let us assume that `Charly` wants to verify σ. After retrieving $A = aP$ the verifying algorithm checks if the equation

$$e(\sigma, P) \stackrel{?}{=} e(H(m), A)$$

holds. Since a valid signature σ fulfills this equation we have

$$\begin{aligned} e(\sigma, P) &= e(H(m), aP) \\ &= e(aH(m), P). \end{aligned}$$

So ensuring $\sigma_m = aH(m)$ yields a valid signature. This is exactly the signing algorithm. Considering that the proxy has $R_{AB} = \frac{b}{a}$, the resigning of a signature σ into $\sigma'$is quite trivial as

$$\sigma'_m = R_{AB} \cdot \sigma_m = bH(m).$$

Now we give an overview of all three algorithms.

ALGORITHM. `Aylin` signs.

Public input: The groups $\mathbf{G}_1$ and $\mathbf{G}_2$, the base point P and its order n, the bilinear pairing $e\colon \mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$ and the hash function $H\colon \{0,1\}^* \to \mathbf{G}_1$.

Input: `Aylin`'s private key $a \in \mathbb{Z}_n^\times$, the message $m \in \{0,1\}^*$.

Output: Signature σ.

1. Compute $\sigma \leftarrow aH(m)$.
2. Return σ.

ALGORITHM. The proxy re-signs.

Public input: The groups $\mathbf{G}_1$ and $\mathbf{G}_2$, the base point P and its order n, the bilinear pairing $e\colon \mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$ and the hash function $H\colon \{0,1\}^* \to \mathbf{G}_1$.

Input: The re-signature key $R_{AB} = \frac{b}{a}$, the message $m \in \{0,1\}^*$ and the signature σ.

Output: σ'.

1. Compute $\sigma' \leftarrow R_{AB} \cdot \sigma$.
2. Return σ'.

ALGORITHM. `Boris` verifies.

Public input: The groups $\mathbf{G}_1$ and $\mathbf{G}_2$, the base point P and its order n, the bilinear pairing $e\colon \mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$ and the hash function $H\colon \{0,1\}^* \to \mathbf{G}_1$.

Input: `Aylin`'s public key A, the message $m \in \{0,1\}^*$ and the signature σ.

Output: {ACCEPT, REJECT}.

1. Compute $U \leftarrow e(H(m), A)$.
2. Compute $V \leftarrow e(\sigma, P)$.
3. If $V = U$ then
4. Return ACCEPT.
5. Else
6. Return REJECT.

5.5.2. Single-use unidirectional scheme. Again fix two groups $\mathbf{G}_1$ and $\mathbf{G}_2$ of prime order n and also generators P and Q for $\mathbf{G}_1$. Choose a non-degenerate bilinear mapping $e\colon \mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$ and a hash function $h\colon \{0,1\}^* \to \mathbb{Z}_n^\times$.

In this scheme we again consider three parties `Aylin`, `Boris` and the proxy. The difference here is that the users have public key pairs instead of just one public key. This means that, for `Aylin` we say that she has a public key pair $A = aP, A' = \frac{1}{a}Q$ where a is her *strong* secret and aQ her *weak* secret. Similarly `Boris` has a public key pair $B = bP, B' = \frac{1}{b}Q$ where b is his *strong* secret and bQ the *weak* one.

Again we assume that the proxy already has the *re-signature key* $R_{AB} = \frac{b}{a}Q$ which enables him to convert signatures of `Aylin` into signatures of `Boris`. Since this scheme allows the proxy to convert a signature only once, we are distinguishing between a lavel 0 signature and a level 1 signature. Now assume that `Charly` wants to verify the level 0 signature $\sigma^{(0)} = (s, R)$ generated by `Aylin` on a message $m \in \{0,1\}^*$. His verification algorithm checks if the equation

$$e(P, sQ) \stackrel{?}{=} e(A, R)e(A, h(m||R)Q) \tag{5.1}$$

holds. Similarly as above we construct $R = rQ$ for a random integer r. Since a valid signature fulfills the equation (5.1) we obtain

$$\begin{aligned} e(P, sQ) &= e(A, R)e(A, h(m||R)Q) \\ &= e(aP, R + h(m||R)Q) \\ &= e(P, a(R + h(m||R)Q)) \\ &= e(P, a(h(m||R) + r)Q). \end{aligned}$$

Now we can see that a level 0 signature can be signed by

$$\sigma^{(0)} = (s, R) = (a(h(m||R) + r), rQ).$$

If the proxy is asked to convert this signature $\sigma^{(0)} = (s, R)$ into $\sigma^{(1)} = (s', R)$ with the re-signature key $R_{AB} = \frac{b}{a}Q$ he computes

$$\sigma^{(1)} \leftarrow (sR_{AB}, R) = (s\frac{b}{a}Q, R) = (b(h(m||R) + r)Q, R).$$

This slightly changes the verification process for a level 1 signature. If `Charly` wants to verify a level 1 signature $\sigma^{(1)} = (S, R)$ with the public key B of `Boris` he checks if the equation

$$e(P, S) \stackrel{?}{=} e(B, R)e(B, H(m||R)Q).$$

holds. Notice that `Aylin` can also directly produce a level 1 signature as

$$\sigma^{(2)} = (a(H(m||R) + r)Q, rQ).$$

The difference between the verification of a level 0 and a level 1 signature can be seen clearly in the overview of all three algorithms.

ALGORITHM. `Aylin` signs.

Public input: The groups $\mathbf{G}_1$ and $\mathbf{G}_2$, the generators of P and Q $\mathbf{G}_1$, the non-degenerate bilinear pairing $e\colon \mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$ and the hash function $h\colon \{0,1\}^* \to \mathbb{Z}_n^\times$.

Input: `Aylin`'s private key $a \in \mathbb{Z}_n^\times$, the message $m \in \{0,1\}^*$, $\ell \in \{0,1\}$ the signing level.

Output: Signature $\sigma^{(\ell)}$.

1. Choose a random $r \xleftarrow{\$} \mathbb{Z}_n^\times$.
2. Compute $R \leftarrow rP$.
3. If $\ell = 0$ then
4. $\quad s \leftarrow a(h(m||R) + r)$.
5. $\quad \sigma^{(0)} \leftarrow (s, R)$.
6. Else if $\ell = 2$ then
7. $\quad S \leftarrow a(h(m||R) + r)Q$.
8. $\quad \sigma^{(1)} \leftarrow (S, R)$.
9. Return $\sigma^{(\ell)}$.

ALGORITHM. `Proxy` Re-signs.

Public input: The groups $\mathbf{G}_1$ and $\mathbf{G}_2$, the generators of P and Q $\mathbf{G}_1$, the non-degenerate bilinear pairing $e\colon \mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$, the hash function $h\colon \{0,1\}^* \to \mathbb{Z}_n^\times$.

Input: The re-signature key $R_{AB} = \frac{b}{a}Q$, the message $m \in \{0,1\}^*$ and a level 0 signature $\sigma^{(0)} = (s, R)$
Output: A level 1 signature $\sigma^{(1)}$.

1. Compute $S \leftarrow sR_{AB}$.
2. Set $\sigma^{(1)} \leftarrow (s', R)$.
3. Return $\sigma^{(1)}$.

ALGORITHM. `Boris` verifies.

Public input: The groups $\mathbf{G}_1$ and $\mathbf{G}_2$, the generators P and Q of $\mathbf{G}_1$, the non-degenerate bilinear pairing $e\colon \mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$, the hash function $h\colon \{0,1\}^* \to \mathbb{Z}_n^\times$.
Input: `Aylin`'s public key A, the message $m \in \{0,1\}^*$ and a signature $\sigma^{(\ell)}$ valid for A.
Output: {ACCEPT, REJECT}.

1. If $\ell = 0$ then
   $u \leftarrow e(P, sQ)$.
2. Else if $\ell = 1$ then
   $u \leftarrow e(P, s)$.
3. Compute $v \leftarrow e(A, R)e(A, h(m||R)Q)$.
4. If $v = u$ then
5. Return ACCEPT.
6. Else
7. Return REJECT.

Note that in both schemes we could require the proxy to verify the input signature before translating it. The security of these schemes was shown in the random oracle model for details see **??**.

The left open challenge was to find a *uni-directional multi-use proxy re-signature scheme*. This in Libert & Vergnaud (2008a). This scheme will be analyzed in detail in this thesis.

5.6. Proxy re-encryption. Similar to *proxy re-signatures* a *proxy re-encryption* scheme allows a semi trusted entity called proxy to translate a ciphertext encrypted with the public key PK_A into a ciphertext encrypted with a distinct other public key PK_B. However the proxy cannot learn anything about the messages encrypted under either key. Also based on the publication of Blaze, Bleumer & Strauss (1998), there have been numerous proposal for *proxy re-encryption schemes* (Ateniese, Fu, Green & Hohenberger 2006; Canetti & Ho-

henberger 2007; Chow, Weng, Yang & Deng 2010; Libert & Vergnaud 2008b). Although all these schemes have very interesting properties and applications, a sophisticated analysis is not within the limits of this thesis. For more information on proxy re-signatures and proxy re-cryptography see Shao (2009).

5.7. Tripartite Diffie-Hellman key exchange. In Joux (2004), another useful application of pairings was introduced. The author suggested a *one round protocol for tripartite Diffie-Hellman* which allows three parties to exchange a session key in just one round. The natural variant of the *Diffie-Hellman key exchange protocol* (Diffie & Hellman 1976) needs two rounds.

Again, let E be an elliptic curve over $\mathbb{F}_q$, $P \in E(\mathbb{F}_q)$ a generator for the group $\mathbf{G}$ and $e\colon \mathbf{G} \times \mathbf{G} \to \mathbb{F}_q^\times$ a non-degenerate bilinear pairing.

The parties `Aylin`, `Boris` and `Charly` want to agree on a session key. Thus, they do the following:

3-Party Diffie-Hellman Protocol. `Aylin`.

Public Input The group $\mathbf{G}$, the base point P and its order n, the bilinear pairing $e\colon \mathbf{G} \times \mathbf{G} \to \mathbb{F}_q^\times$.
Input: `Aylin`'s private key a, B public key of `Boris`, C public key of `Charly`.
Output: Session key K.

1. Compute $K \leftarrow e(B, C)^a$.
2. Return K.

3-Party Diffie-Hellman Protocol. `Boris`.

Public Input The group $\mathbf{G}$, the base point P and its order n, the bilinear pairing $e\colon \mathbf{G} \times \mathbf{G} \to \mathbb{F}_q^\times$.
Input: `Boris`' private key b, A public key of `Aylin`, C public key of `Charly`.
Output: Session key K.

1. Compute $K \leftarrow e(A, C)^b$.
2. Return K.

3-Party Diffie-Hellman Protocol. `Charly`.

Public Input The group $\mathbf{G}$, the base point P and its order n, the bilinear pairing $e\colon \mathbf{G} \times \mathbf{G} \to \mathbb{F}_q^\times$.
Input: `Charly`'s private key c, A public key of `Aylin`, B public key of `Boris`.
Output: Session key K.

1. Compute $K \leftarrow e(A, B)^c$.
2. Return K.

In the end all parties have the session key $K = e(P, P)^{abc}$. Note we need a pairing with $e(P, P) \neq 1$. Naturally we require that the *discrete logarithm problem* is both hard in the group $\mathtt{G}$ and in $\mathbb{F}_q^\times$, a detailed security discussion can be found in the original publication. Note also that despite this looks somehow nice on paper, in practice it is not really efficient because of the nature of pairings. Since they map points to elements of a multiplicative field, one has to increase the size of the field for security which again effects the computation of a pairing. The current communication speed on digital channels neutralizes the "gain" of this key exchange scheme.

5.8. Other uses of pairings. The last example above shows that pairings are not only interesting in digital signatures. There are numerous other cryptographic settings where pairings are used. Some very interesting topics in pairing based cryptography are:

- *Identity based cryptography*
- *Authentication*
- *Threshold cryptosystems*
- *Traitor tracing*
- *Hierarchical cryptosystems*
- ...

The *pairing based crypto lounge* Barreto (2009) provides an excellent resource for further information and research on areas based on pairings.

Part II
The Signature Scheme

Suppose that you are the holder of a document m which has been issued and certified in the form of a signature by a specific domain `A`. After a while you want or need to change your membership to another domain `B` and want to take your documents with you without loosing their originality. For example, this can be the true for *digital rights management* (DRM) systems, or for public key certificates validated by different *certification authorities* (CA), or even for future e-passport systems. In short, the valid signature $\sigma_A(m)$ which ensures the authenticity of the document m for domain `A` has to be somehow changed into $\sigma_B(m)$, a valid signature on the same document m for the domain `B`. The detailed form of these cases will be discussed in Part V.

6. The trivial solutions

The first trivial solution to the problem above would be removing the signature of `Aylin` (owner of domain `A`) from the document and replacing it with a signature of `Boris` (owner of domain `B`). This solution, however, is not acceptable since the document can be altered during the re-signing process. This solution would also require interaction between domains and the document holders, may be even for terabytes of data. Considering that there are hundreds of thousands of users, an attacker could expose `Aylin`'s and maybe also `Boris`' secret.

The second trivial solution for changing the signature from a specific domain to another would be, that the holder `Oscar` of the document m, asks `Boris` to *over-sign* the document m with the signature of `Aylin` on it. This means that `Boris` just adds his signature on the document as depicted in Figure 6.1. However, this simple solution has significant disadvantages. First of all `Boris` must either sign *all* the documents himself or he has to delegate his signing rights to some other entities. For `Boris` this means the following:

- It is *very* inefficient. For example in the case of the DRM platforms, there are *many* users and *many* files, which would mean that `Boris` has to *verify* the authenticity of terabytes of data and *over-sign* them all.
- This could also be a security risk, since attackers could impose as false domains and attack `Boris` or his delegatees during signing processes.

On the other hand this means for `Oscar`:

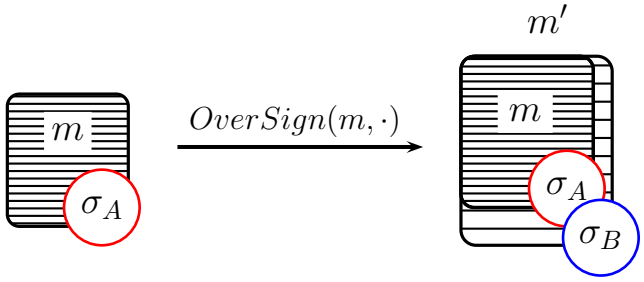


Figure 6.1: Trivial $OverSign()$ process

- `Boris` knows directly that `Oscar` has his document from `Aylin` which compromises `Oscar`'s privacy.
- Each translation causes the actual document m to grow since the new document m' would also include the signature of `Aylin`. Using a short signature of Boneh *et al.* (2004) as introduced in Part I, the signature of `Aylin` on m is $\sigma_A(m) = aH(m)$ then `Boris` over-signs it as $\sigma_B(m') = bH(m||\sigma_A(m))$. The new document would be $m' = m||\sigma_A(m)$ which has the signature $\sigma_A(m)$ of `Aylin` appended to the original document m.

Let us also consider another practical case, where in a company users are separated into independent working groups and each one of them is mandated by a supervisor. The outcome of a project of some cooperating working groups has to be signed by the private key of the company. The trivial solutions would reveal the internal structure of the company as well as the working groups. This is clearly not in the best interest of the company.

7. Requirements

We conclude here that for some application areas as mentioned above and discussed later in Part V, the trivial solutions are not appropriate. Thus, the amount of data and the unavailability of the domain owners' private keys sk_A and sk_B, another entity is required to *translate* a signature from one domain (`Aylin`) into another (`Boris`). We require that:

1. This entity, called, *proxy* is only *semi-trusted* and the information (the re-signature key) that he is granted is limited. This means that a corrupted proxy cannot expose secrets of the domain owners.

2. **Uni-directional, multi-use.** The re-signature key, available to the proxy allows him only to translate signatures from `Aylin` to `Boris` and *not* vice versa.

3. **Private proxy.** The re-signature keys available to the proxy can be kept secret.

4. **Non-transitivity**. The re-signature key R_{AB} which allows the proxy to translate signatures from `Aylin` to `Boris` and the re-signature key R_{BC} which allows the proxy to translate signatures from `Boris` to `Charly`, does not give him the ability to calculate the re-signature key R_{AC} for translating signatures from `Aylin` to `Charly` directly.

5. **Non-interactive**. The re-signature key R_{AB} can be calculated without the interaction of `Aylin`.

6. **Transparency**. Users do not need to know that a proxy translated the signature.

7. **Unlinkability**. The translated documents can not be linked to the previous signer `Aylin`, ie. the new signature has no connection with `Aylin` and it is a perfectly valid signature of `Boris`.

8. **Key optimal**. The domain owners do not need to store more information than their secret key sk.

Obviously we also need that this signature scheme is secure against chosen message attacks, desirably in the *standard model*. This requires a *bullet proof* security definition and proof of the security which we discuss in Part III. Now we introduce the *multi-use uni-directional proxy re-signature* scheme from Libert & Vergnaud (2008a), step by step, for a comprehensive understanding.

8. A multi-use uni-directional proxy re-signature

As in Part I, instead of writing down all the formal definitions, we will try to explore this signature scheme step by step for a clear understanding. Assume that we have a signature $\sigma_A(m)$ from an entity `Aylin` on a document m and we want this document to be authentic for another entity `Boris`. Since at this point this signature $\sigma_A(m)$ can be an arbitrary signature scheme, we remember the *short signature* from Boneh, Lynn & Shacham (2004) introduced in Part I. It is safe and also practical to assume that we have a *short signature* of `Aylin` on the message m.

More formally, for a generator P of the group $\mathbf{G}$, a hash function $H\colon \{0,1\}^* \to \mathbf{G}$ and a non-degenerate bilinear mapping $e\colon \mathbf{G} \times \mathbf{G} \to \mathbb{Z}_p^\times$:

- The public and private key pair of `Aylin` is (X_A, x_A) with $x_A \in \mathbb{Z}_p^\times$ and $X_A = x_A P$.
- The valid signature $\sigma_A(m)$ of `Aylin` on $m \in \{0,1\}^*$ is computed as $\sigma_A(m) = \sigma_0 = x_A H(m)$.
- A signature is verified by checking $e(\sigma_0, P) \stackrel{?}{=} e(H(m), X_A)$.

Note that as in every signature scheme the public key is needed for verification. This means that the short signature is actually a four tuple $\sigma_A(m) = (x_A H(m), X_A, H(m), P)$ with the public key X_A, the hash value of the message $H(m)$ and the generator P of the group $\mathbf{G}$. To understand the relation of these 4 elements, we graphically connect them to a big "H" as shown on the left side of Figure 8.1.

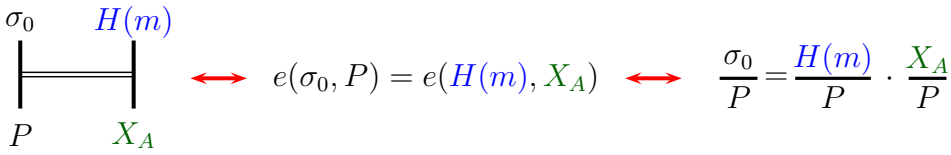


Figure 8.1: The H-representation

This H tells us that, pairing of the signature σ_0 and the base point P equals the pairing of the hash value $H(m)$ of the message and the public key X_A, depicted in the middle of Figure 8.1. Equivalently the discrete logarithm to base P of σ_0 is equal to the discrete logarithm of $H(m)$ to base P *times* the discrete logarithm of X_A to base P. This is expressed in a slightly unusual discrete logarithm notation on the right hand side of Figure 8.1.

Now we want the proxy to *re-sign* this signature into one which is valid for the public key $X_B = x_B P$ of `Boris`. To do so, the proxy has to blind out the x_A in the signature to assure unlinkability to `Aylin` and use the information that he has, namely the re-signature key, in constructing the translated signature appropriately. We want the signature

$$\frac{\sigma_0}{P} = \frac{H(m)}{P} \cdot \left(\frac{X_A}{P}\right) \tag{8.1}$$

to be changed into another signature like

$$\frac{\overline{\sigma}_0}{P} = \frac{H(m)}{P} \cdot \left(\frac{\overline{\sigma}_1}{P}\right),$$

where the encircled element of equation (8.1) is changed into an element $\frac{\overline{\sigma}_1}{P}$. Since $\overline{\sigma}_1$ cannot be X_B because for that transformation the private key of `Aylin` is needed, it has to fulfill another relation like

$$\frac{\overline{\sigma}_1}{P} = \frac{\overline{\sigma}_{-1}}{P} \cdot \frac{X_B}{P}.$$

The translated signature has three elements $(\overline{\sigma}_0, \overline{\sigma}_1, \overline{\sigma}_{-1})$ and is related to the public key X_B of `Boris`. They fulfill the following relations:

$$\frac{\overline{\sigma}_0}{P} = \frac{H(m)}{P} \cdot \frac{\overline{\sigma}_1}{P},$$

$$\frac{\overline{\sigma}_1}{P} = \frac{\overline{\sigma}_{-1}}{P} \cdot \frac{X_B}{P}.$$

Converting these discrete logarithm relations into pairing equations we get

$$e(\overline{\sigma}_0, P) = e(H(m), \overline{\sigma}_1),$$
$$e(\overline{\sigma}_1, P) = e(\overline{\sigma}_{-1}, X_B).$$

Graphically speaking, from these two equations we get two H s which are connected to each other. This means that the proxy extended the signature by adding one H with two new elements to the original H as shown in Figure 8.2.

Now we want to determine the relations between the elements of the original signature and the elements of the translated signature to understand the translation process. First, we know that the public key of `Boris` is $X_B = x_B P$ and $\overline{\sigma}_{-1} = tP$ where $x_B, t \in \mathbb{Z}_p^\times$. Thus, to fulfill the relations from above we have $\overline{\sigma}_1 = t x_B P$ and $\overline{\sigma}_0 = t x_B H(m)$. Consider Figure 8.3, the original signature of `Aylin` on the left side is translated into a signature of `Boris` on the right hand side.

Since t cannot be $\frac{x_A}{x_B}$ because the proxy is only semi trusted we know that $t = r \cdot \frac{x_A}{x_B}$ for some $r \xleftarrow{\$} \mathbb{Z}_p^\times$. Therefore $\overline{\sigma}_1 = r X_A$ and $\overline{\sigma}_{-1} = r \cdot \frac{x_A}{x_B} P$. Now we can clearly see what happens in the translation process, the proxy, on receiving a *valid* signature $\sigma = x_A H(m)$ for the public key $X_A = x_A P$, chooses a random $r \xleftarrow{\$} \mathbb{Z}_p^\times$ and re-signs the signature as:

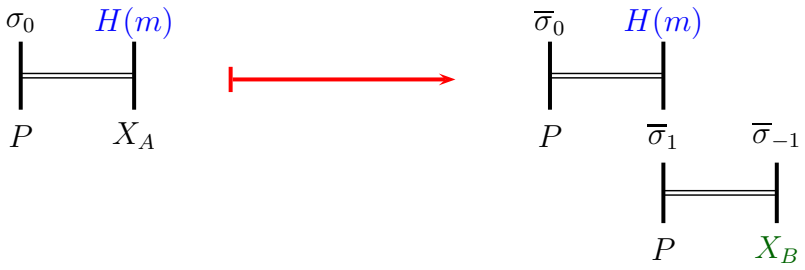


Figure 8.2: Extending the "H".

$$\sigma_0 = x_A \cdot H(m) \qquad \overline{\sigma}_0 = t \cdot x_B \cdot H(m)$$
$$X_A = x_A \cdot P \qquad \overline{\sigma}_1 = t \cdot x_B \cdot P$$
$$\overline{\sigma}_{-1} = t \cdot P$$
$$X_B = x_B \cdot P$$

Figure 8.3: Translation of the Signature $x_A H(m)$

$$\overline{\sigma} = (\overline{\sigma}_0, \overline{\sigma}_1, \overline{\sigma}_{-1}) = (r \cdot \sigma_0, r \cdot X_A, r \cdot \tfrac{x_A}{x_B} P) = (r \cdot x_A H(m), r \cdot x_A P, r \cdot R_{AB}).$$

Setting $t = r\frac{x_A}{x_B}$ yields:

$$\overline{\sigma} = (\overline{\sigma}_0, \overline{\sigma}_1, \overline{\sigma}_{-1}) = (t x_B H(m), t x_B P, tP). \tag{8.2}$$

One verifies the signature by checking the two H relations:

$$e(\overline{\sigma}_0, P) \stackrel{?}{=} e(H(m), \overline{\sigma}_1) \qquad \wedge \qquad e(\overline{\sigma}_1, P) \stackrel{?}{=} e(\overline{\sigma}_{-1}, X_B).$$

Note that we assume that the proxy already has the re-signature key which is $R_{AB} = \frac{x_A}{x_B} P$. This re-signature key can be calculated as $(x_B)^{-1} X_A = \frac{x_A}{x_B} P$ and given to him by `Boris` without interacting with `Aylin` (*non-interactivity*).

Note also that R_{AB} allows the proxy to convert signatures only from `Aylin` to `Boris` and not the other way around (*uni-directionality*).

In equation (8.2) we see that `Boris` can also construct this signature by himself. This means that one cannot distinguish between a translated signature and signature which was signed like that (*transparency*).

The idea behind this is to exploit the Diffie-Hellman assumption that given tP for some $t \in \mathbb{Z}_p^\times$ it is hard to generate tx_BP without knowing the secret $x_B \in \mathbb{Z}_p^\times$ of `Boris`. The valid short signature $\sigma = x_A H(m)$ is re-randomized and blinded into $\overline{\sigma} = (tx_B H(m), tx_B P, tP)$ with a random element $t \in \mathbb{Z}_p^\times$.

Now we want use this idea of *re-randomizing* and *blinding* by adding two new elements iteratively to extend the translation process into a *multi-use* scheme. To obviate confusion we call the short signature of Boneh *et al.* (2004) a level 0 signature and the translated one a level 1 signature, thus a signature which was translated ℓ times, will be called a level ℓ signature. Now consider the level ℓ signature with $2\ell+1$ elements valid for the public key X_i represented in the "H" form shown in Figure 8.4.

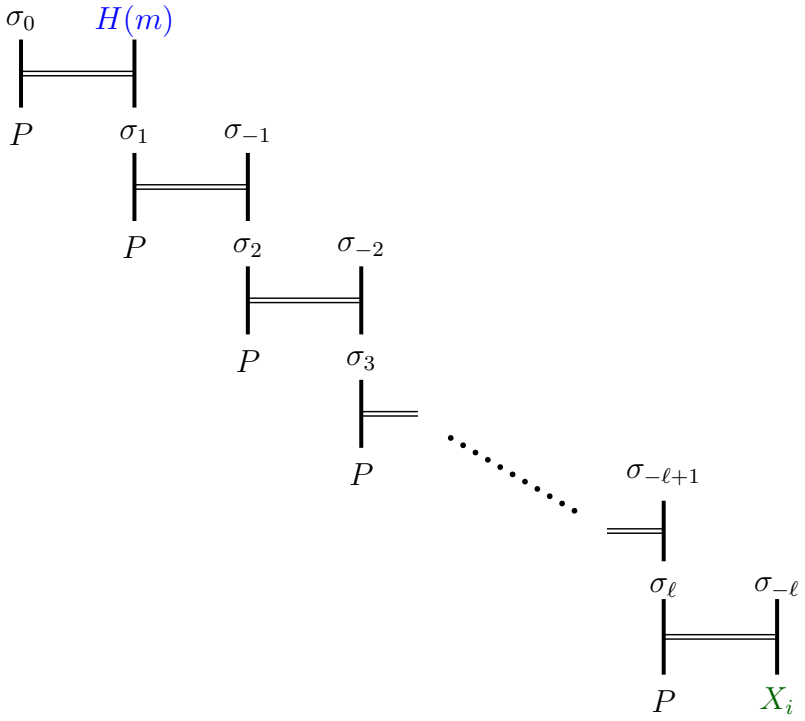


Figure 8.4: An level ℓ signature

As we know each H corresponds to one bilinear verification equation as $e(\cdot,\cdot) \stackrel{?}{=} e(\cdot,\cdot)$, counting a total of $\ell + 1$ for a level ℓ signature. Using the discrete logarithm notation from above we observe that for a valid signature

the following equations hold:

$$\frac{\sigma_0}{P} = \frac{H(m)}{P} \cdot \frac{\sigma_1}{P}, \qquad \frac{\sigma_1}{P} = \frac{\sigma_{-1}}{P} \cdot \frac{\sigma_2}{P}, \qquad \frac{\sigma_2}{P} = \frac{\sigma_{-2}}{P} \cdot \frac{\sigma_3}{P}, \qquad \ddots$$

$$\frac{\sigma_\ell}{P} = \frac{\sigma_{-\ell}}{P} \cdot \frac{X_i}{P}.$$

We also observe that, all these equations are connected to each other like *chains*. For example in the first equation we can replace $\frac{\sigma_1}{P}$ using the second equation to obtain

$$\frac{\sigma_0}{P} = \frac{H(m)}{P} \cdot \frac{\sigma_{-1}}{P} \cdot \left(\frac{\sigma_2}{P}\right).$$

Repeating this for all equations results in $\ell + 1$ equations

$$(0) \qquad \frac{\sigma_0}{P} = \frac{H(m)}{P} \cdot \frac{\sigma_{-1}}{P} \cdot \frac{\sigma_{-2}}{P} \frac{\sigma_{-3}}{P} \cdot \cdots \cdot \frac{\sigma_{-\ell+1}}{P} \cdot \frac{\sigma_{-\ell}}{P} \cdot \frac{X_i}{P},$$

$$(1) \qquad \frac{\sigma_1}{P} = \frac{\sigma_{-1}}{P} \cdot \frac{\sigma_{-2}}{P} \cdot \cdots \cdot \frac{\sigma_{-\ell}}{P} \cdot \frac{X_i}{P},$$

$$(2) \qquad \frac{\sigma_2}{P} = \frac{\sigma_{-2}}{P} \cdot \frac{\sigma_{-3}}{P} \cdot \cdots \cdot \frac{\sigma_{-\ell}}{P} \cdot \frac{X_i}{P},$$

$$(3) \qquad \frac{\sigma_3}{P} = \frac{\sigma_{-3}}{P} \cdot \frac{\sigma_{-4}}{P} \cdot \cdots \cdot \frac{\sigma_{-\ell}}{P} \cdot \frac{X_i}{P},$$

$$(\vdots) \qquad \vdots \quad \vdots$$

$$(\ell - 1) \qquad \frac{\sigma_{\ell-1}}{P} = \frac{\sigma_{-\ell+1}}{P} \cdot \frac{\sigma_{-\ell}}{P} \cdot \frac{X_i}{P},$$

$$(\ell) \qquad \frac{\sigma_\ell}{P} = \frac{\sigma_{-\ell}}{P} \cdot \frac{X_i}{P}.$$

Now we want to interpret these equations. We know that $X_i = x_i P$ and we write the elements $\sigma_{-k} = r_k P$ for $k \in \{1, \ldots, \ell\}$ with $r_k \in \mathbb{Z}_p^\times$. This means that

$$\sigma_{-\ell} = r_\ell P, \quad \sigma_{-\ell+1} = r_{\ell-1} P, \quad \cdots, \quad \sigma_{-2} = r_2 P, \quad \sigma_{-1} = r_1 P.$$

One can verify the valid level ℓ signature $\sigma^{(\ell)}(m)$ on message m for the public key X_i by checking the following $\ell + 1$ equations:

$$(0) \qquad e(\sigma_0, P) \stackrel{?}{=} e(H(m), \sigma_1),$$

$$(k) \qquad e(\sigma_k, P) \stackrel{?}{=} e(\sigma_{k+1}, \sigma_{-k}) \quad k \in \{1, \ldots, \ell - 1\},$$

$$(\ell) \qquad e(\sigma_\ell, P) \stackrel{?}{=} e(\sigma_{-\ell}, X_i).$$

Thus a level ℓ signature valid for the public key X_i has the form as in Figure 8.5 which also shows the *signing* process at level ℓ. In short ℓ random coefficients $r_k \xleftarrow{\$} \mathbb{Z}_p^\times$ for $k \in \{1, ..., \ell\}$ are chosen and multiplied as in Figure 8.5.

$$\sigma_0^{(\ell)} = (r_\ell \cdots r_1) x_i H(m),$$

$$\begin{aligned}
\sigma_1^{(\ell)} &= (r_\ell \cdot \cdots \cdot \cdot \cdot r_1) x_i P, & \sigma_{-1}^{(\ell)} &= r_1 P, \\
\sigma_2^{(\ell)} &= (r_\ell \cdot \cdots \cdot r_2) x_i P, & \sigma_{-2}^{(\ell)} &= r_2 P, \\
\sigma_3^{(\ell)} &= (r_1 \cdots r_3) x_i P, & \sigma_{-3}^{(\ell)} &= r_3 P, \\
&\vdots & &\vdots \\
\sigma_\ell^{(\ell)} &= r_\ell x_i P, & \sigma_{-\ell}^{(\ell)} &= r_\ell P.
\end{aligned}$$

Figure 8.5: Coefficient Representation

To understand the translation process, we now want the proxy to translate this signature into a level $\ell + 1$ signature valid for the public key $X_j = x_j P$ of user j. Similar to Figure 8.2, the proxy has to add one H at the end of Figure 8.4 for blinding out the public key X_i. We assume that user j has already

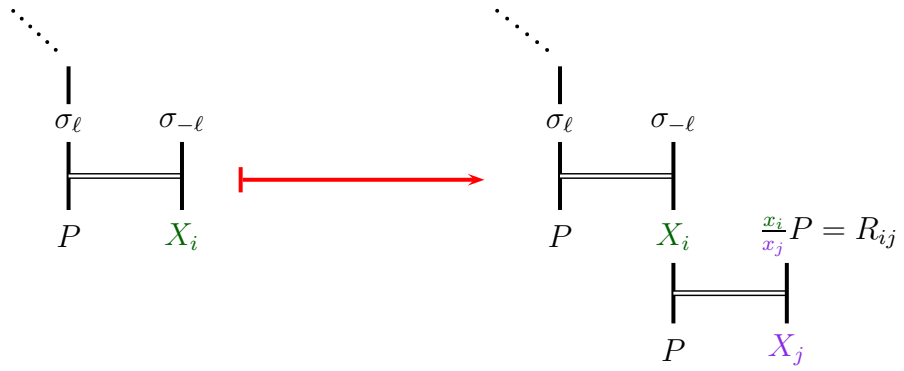


Figure 8.6: Appending one H to a level ℓ signature

delegated the re-signature key $R_{ij} = \frac{x_i}{x_j}P = x_j^{-1}X_i$ to the proxy. As before the proxy adds one H at the end of the signature as shown in Figure 8.6. The resulting signature is perfectly valid for X_j the public key of user j. However, the elements X_i and R_{ij} are visible and allow an attacker extract to the re-signature key from it. Thus the proxy wants to blind out the re-signature key R_{ij} and the public key X_i of user i. To do so, the proxy choses a random $t_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p^\times$ and re-randomizes X_i and R_{ij} as $t_{\ell+1}X_i$ and $t_{\ell+1}R_{ij}$ respectively.

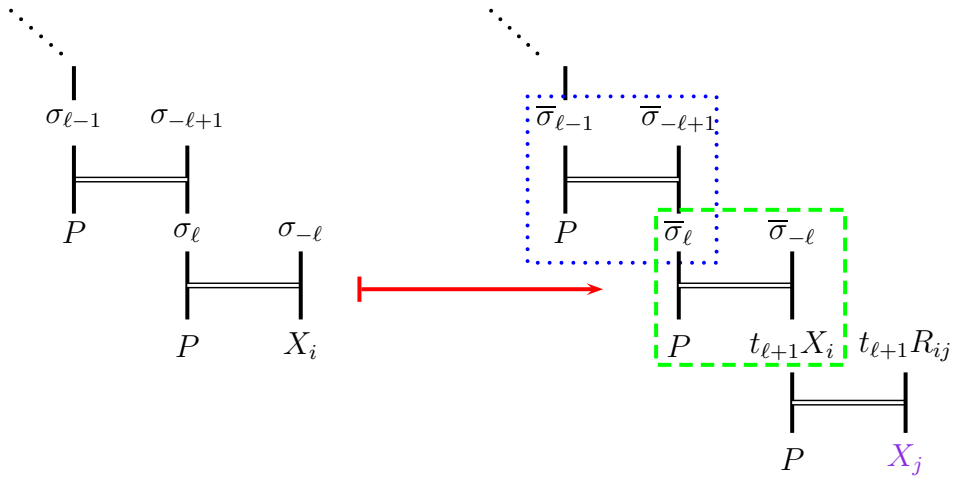


Figure 8.7: The re-randomization of the level $\ell + 1$ signature

Now consider Figure 8.7, we observe that the equation of the H inside the (green) dashed frame will not hold if $\overline{\sigma}_\ell = \sigma_\ell$ and $\overline{\sigma}_{-\ell} = \sigma_{-\ell}$. Therefore the proxy multiplies σ_ℓ with $t_{\ell+1}$ such that $\overline{\sigma}_\ell = t_{\ell+1}\sigma_\ell$. Since all the H-s are connected to each other the multiplication $\overline{\sigma}_\ell = t_{\ell+1}\sigma_\ell$, clearly breaks the equations of the upper H-s. To reassure the equations of each H the multiplication with $t_{\ell+1}$ has to ripple all the way up to $\overline{\sigma}_0 = t_{\ell+1}\sigma_0$.

However this is not the only problem which occurs while translating the signature. Since at this point the proxy did not do anything to $\overline{\sigma}_{-k}$ for $k \in \{1, \ldots, \ell\}$, the signature is easily linkable to its predecessor because $\overline{\sigma}_{-\ell} = \sigma_{-\ell}$, $\overline{\sigma}_{-\ell+1} = \sigma_{-\ell+1}$,..., $\overline{\sigma}_{-1} = \sigma_{-1}$. Thus, the proxy chooses another random coefficient $t_\ell \xleftarrow{\$} \mathbb{Z}_p^\times$ and *re-randomizes* $\sigma_{-\ell}$ as $\overline{\sigma}_{-\ell} = t_\ell\sigma_{-\ell}$. Again the relation of the H inside the (green) dashed frame is broken. So, the proxy has to multiply $\overline{\sigma}_\ell$ with t_ℓ to reassure the integrity of this H and since we have $\overline{\sigma}_\ell = t_{\ell+1}t_\ell\sigma_\ell$ the multiplication with t_ℓ also has to ripple all the way up to $\overline{\sigma}_0$ which is now $\overline{\sigma}_0 = t_{\ell+1}t_\ell\sigma_0$.

Now consider the H inside the (blue) dotted frame, we have $\overline{\sigma}_\ell = t_{\ell+1}t_\ell\sigma_\ell$, also to avoid linkability here, the proxy chooses another random coefficient $t_{\ell-1} \xleftarrow{\$} \mathbb{Z}_p^\times$ and multiplies $\sigma_{-\ell+1}$ with it which means that $\overline{\sigma}_{-\ell+1} = t_{\ell-1}\sigma_{-\ell+1}$. The multiplication $\overline{\sigma}_{\ell-1} = t_{\ell+1}t_\ell t_{\ell-1}\sigma_{\ell-1}$ then reassures that the equation of the H inside the (blue) dotted frame holds. But to reassure the integrity of the upper H-s this multiplication with $t_{\ell-1}$ has also to ripple all the way up to $\overline{\sigma}_0$ which is then $\overline{\sigma}_0 = t_{\ell+1}t_\ell t_{\ell-1}\sigma_0$. Following this process up to the top, we get the translated level $\ell + 1$ signature as

$$\overline{\sigma}_0 = (t_{\ell+1} \cdots t_1)\sigma_0$$

$$\begin{aligned}
\overline{\sigma}_1 &= (t_{\ell+1} \cdot \cdots \cdot t_1)\sigma_1, & \overline{\sigma}_{-1} &= t_1\sigma_{-1},\\
\overline{\sigma}_2 &= (t_{\ell+1} \cdot \cdots \cdot t_2)\sigma_2, & \overline{\sigma}_{-2} &= t_2\sigma_{-2},\\
\overline{\sigma}_3 &= (t_{\ell+1} \cdots t_3)\sigma_3, & \overline{\sigma}_{-3} &= t_3\sigma_{-3},\\
&\vdots & &\vdots\\
\overline{\sigma}_\ell &= t_{\ell+1}t_\ell\sigma_\ell, & \overline{\sigma}_{-\ell} &= t_\ell\sigma_{-\ell},\\
\overline{\sigma}_{\ell+1} &= t_{\ell+1}X_i, & \overline{\sigma}_{-\ell-1} &= t_{\ell+1}R_{ij}.
\end{aligned}$$

Setting $\tilde{r}_{\ell+1} = t_{\ell+1}\frac{x_i}{x_j}$ and $\tilde{r}_k = t_k r_k$ for $k \in \{1, \ldots, \ell\}$ gives us similar to Figure 8.5 that the level $\ell + 1$ signature is valid for the public key X_j on the

same message m

$$\sigma_{X_j}^{(\ell+1)}(m) = (\sigma_0^{(\ell+1)}, \quad \sigma_1^{(\ell+1)}, \quad \cdots \quad , \sigma_{\ell+1}^{(\ell+1)}, \quad \sigma_{-\ell-1}^{(\ell+1)}, \quad \cdots \quad , \sigma_{-1}^{(\ell+1)})$$

with $2\ell + 3$ elements as:

$$\sigma_0^{(\ell+1)} = (\tilde{r}_{\ell+1} \cdots \tilde{r}_1) x_j H(m),$$

$$\begin{aligned}
\sigma_1^{(\ell+1)} &= (\tilde{r}_{\ell+1} \cdot \cdots \cdot \tilde{r}_1) x_j P, & \sigma_{-1}^{(\ell+1)} &= \tilde{r}_1 P, \\
\sigma_2^{(\ell+1)} &= (\tilde{r}_{\ell+1} \cdot \cdots \cdot \tilde{r}_2) x_j P, & \sigma_{-2}^{(\ell+1)} &= \tilde{r}_2 P, \\
\sigma_3^{(\ell+1)} &= (\tilde{r}_{\ell+1} \cdots \tilde{r}_3) x_j P, & \sigma_{-3}^{(\ell+1)} &= \tilde{r}_3 P, \\
&\vdots & &\vdots \\
\sigma_\ell^{(\ell+1)} &= \tilde{r}_{\ell+1} \tilde{r}_\ell x_j P, & \sigma_{-\ell}^{(\ell+1)} &= \tilde{r}_\ell P, \\
\sigma_{\ell+1}^{(\ell+1)} &= \tilde{r}_{\ell+1} x_j P, & \sigma_{-\ell-1}^{(\ell+1)} &= \tilde{r}_{\ell+1} P.
\end{aligned}$$

Note that we already explained the relation between the verification equations and the elements of the signature with the H notation. In the next section we define a set of tools which help us to understand the signing an re-signing processes explained above.

9. Building blocks of signature scheme

In this section, we decompose the signature scheme into simple building blocks. These simple building blocks will make it easier to understand the following sections.

9.1. The building blocks. Recalling the previous section, we can divide the building blocks into two main categories. In the first category we have the building blocks which lengthen the signature. In the second category we have the building blocks which randomize the signature elements.

9.1.1. Lengthening the signature. We can extend a level ℓ signature by one H in two different ways. Namely, by adding a trivial H or by adding a re-signature H, which are both explained below.

Building block ADD TRIVIAL H . This building block on input a level ℓ signature valid for the public key X_i extends the signature by one H as shown in Figure 9.1. Despite the redundancy of the last H, the result is a level $\ell + 1$

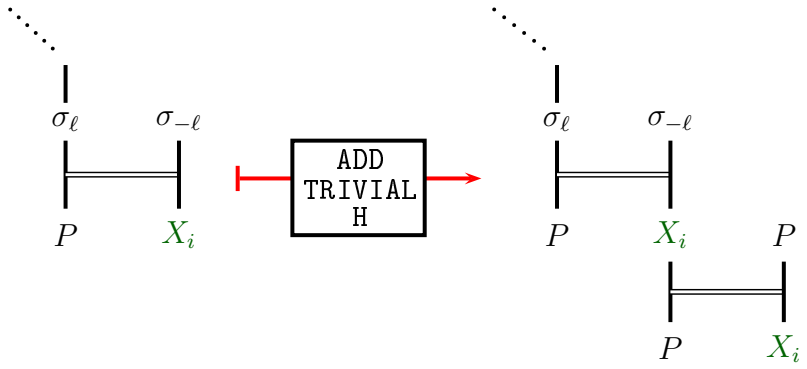


Figure 9.1: Adding a trivial H to a level ℓ signature

signature still perfectly valid for the public key X_i.

Building block ADD RE-SIGN H . This building block on input a level ℓ signature valid for the public key X_i, a re-signature key $R_{ij} = \frac{x_i}{x_j}P$ and another public key X_j extends the signature by one H as shown in Figure 9.7. The

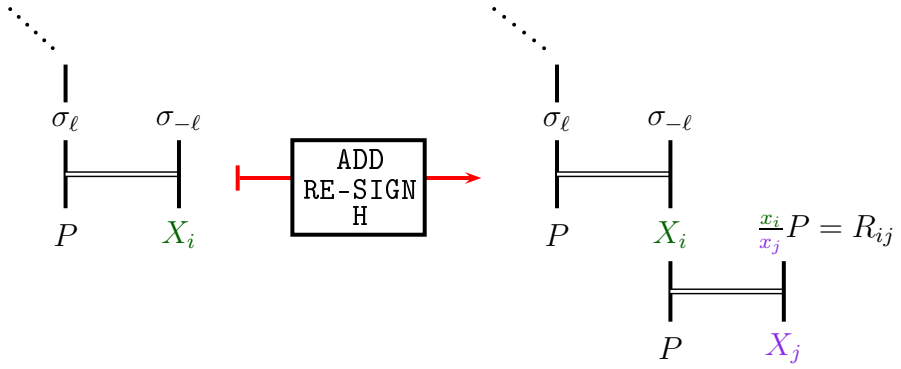


Figure 9.2: Adding a re-signature H to a level ℓ signature

resulting level $\ell + 1$ signature is now valid for the public key X_j.

9.1.2. Randomization of the signature elements. Recall the previous section where we observed that a multiplication of a signature element ripples all the way up to reassure the integrity of each upper H. We start with the introduction of the building block RE-RANDOM i which starts randomizing at a certain height i of the input signature with only one coefficient.

Building block RE-RANDOM i . This building block on input a level $\ell \geq i$ signature chooses a random coefficient $r_i \xleftarrow{\$} \mathbb{Z}_p^\times$ and multiplies the elements $\sigma_0, \ldots, \sigma_i$ and σ_{-i} with r_i as shown in Figure 9.3. The resulting new signature

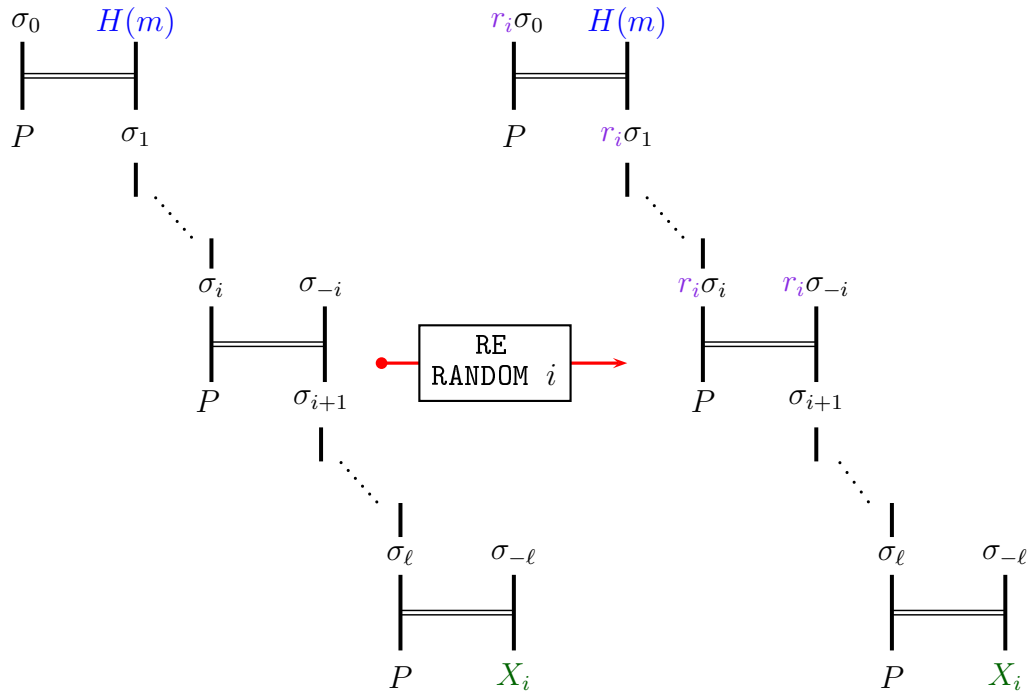


Figure 9.3: Re-randomizing $\sigma^{(\ell)}$ at height i

is still perfectly valid for the public key X_i.

Building block RE-RANDOM . This building block on input a level ℓ signature chooses ℓ random coefficients $r_1, \ldots, r_\ell \xleftarrow{\$} \mathbb{Z}_p^\times$ and multiplies them as shown in Figure 9.4. Note that RE-RANDOM $= \prod_{i=1}^{\ell}$ RE-RANDOM i where

the product means that all RE-RANDOM i are applied one after another. The resulting signature is still perfectly valid for the public key X_i.

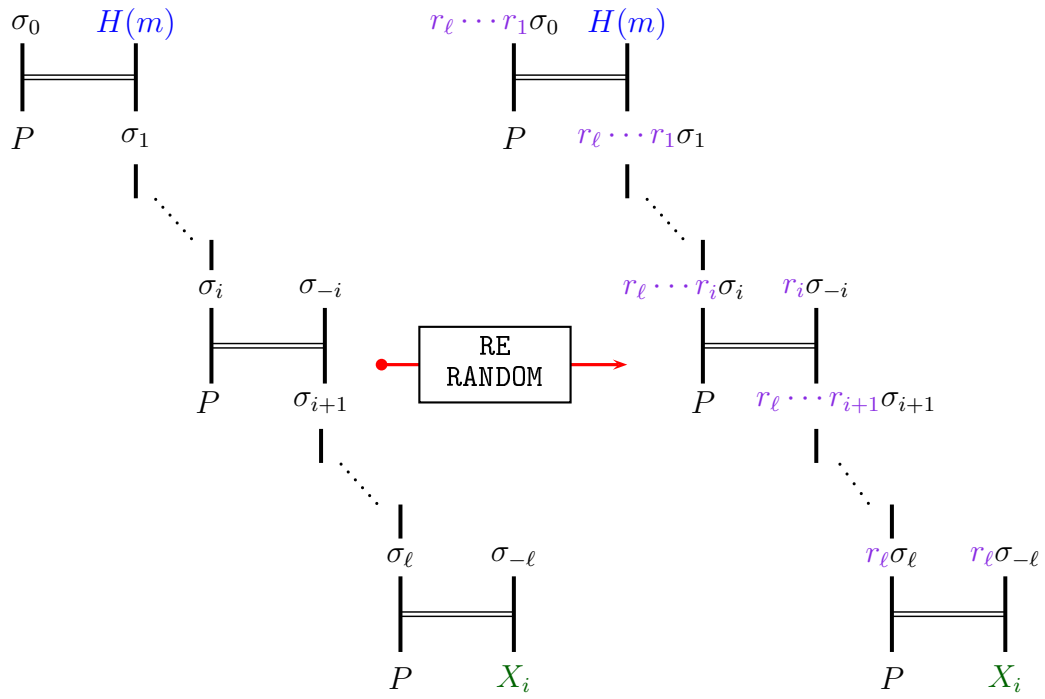


Figure 9.4: Re-randomizing $\sigma^{(\ell)}$ completely

We observe that lengthening and re-randomizing the signature is easy. Now we want to explain the signing and re-singing processes using these building blocks.

9.2. Signing at level ℓ. Recall that a level 0 signature $\sigma^{(0)}$ valid for the public key $x_A P = X_A$ of user A in the H representation looks like Figure 9.5. After computing this $\sigma^{(0)}$ user A uses the building block ADD TRIVIAL H on $\sigma^{(0)}$ ℓ times and obtains an extended level 0 signature as depicted on the left side of Figure 9.6. User A then uses building block RE-RANDOM on this extended signature and obtains $\sigma^{(\ell)}$ which is depicted on the right hand side of Figure 9.6.

Summarizing this, a user A with public key X_A first computes a level 0 signature $\sigma^{(0)}$ and uses the building block ADD TRIVIAL H ℓ times to lengthen the signature. Using the building block RE-RANDOM on this extended signature

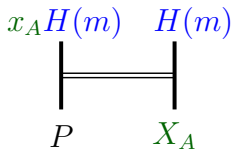


Figure 9.5: A level 0 signature

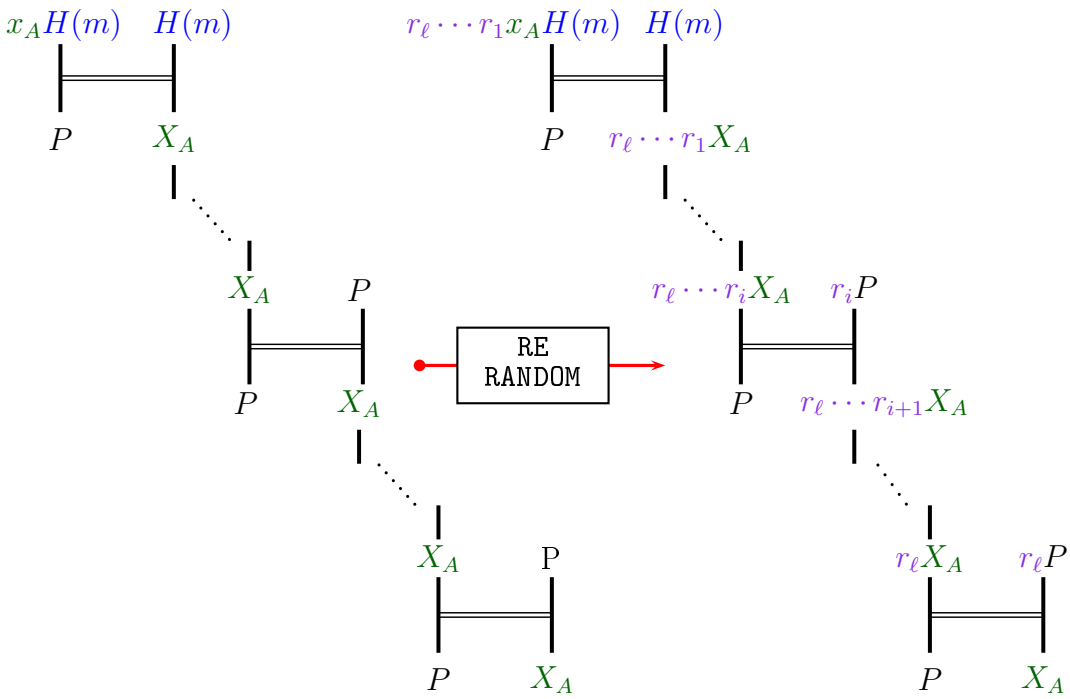


Figure 9.6: Re-randomizing the extended $\sigma^{(0)}$

gives A proper a level ℓ signature.

9.3. Re-signing a level ℓ signature. Now suppose that a proxy is asked to re-sign the level ℓ signature valid for the public key X_A of user A from above into a level $\ell + 1$ signature valid for the public key X_B of user B. Assuming that user B already delegated the re-signature key $\frac{x_A}{x_B} P = R_{AB}$ to the proxy, the re-signing process can be explained in two steps.

Step 1. The proxy uses the building block `ADD RE-SIGN H` to append a new H consisting of the re-signature key R_{AB} and the new public key X_B to the

level ℓ signature from above as shown in Figure 9.7. The result is a level $\ell + 1$

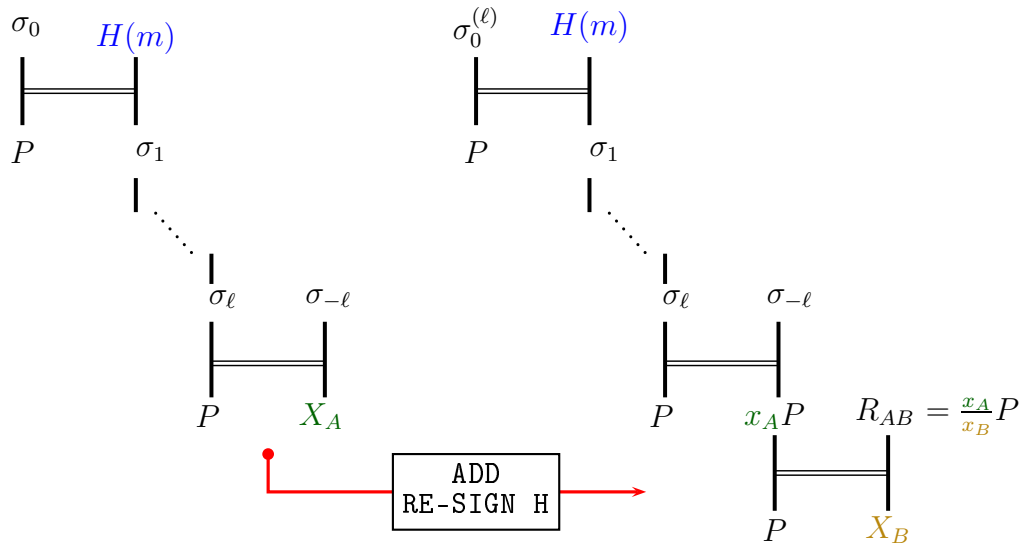


Figure 9.7: Step 1: adding the re-sign H

signature perfectly valid for the public key X_B.

Step 2. Now the proxy uses RE-RANDOM to re-randomize the signature and blind out the elements. As shown in Figure 9.8 RE-RANDOM chooses $\ell + 1$ random coefficients $(r_{\ell+1}, \ldots, r_1) \xleftarrow{R} \mathbb{Z}_p^\times$ and multiplies them accordingly. Summarizing the results from above we now write down everything in a formal notation.

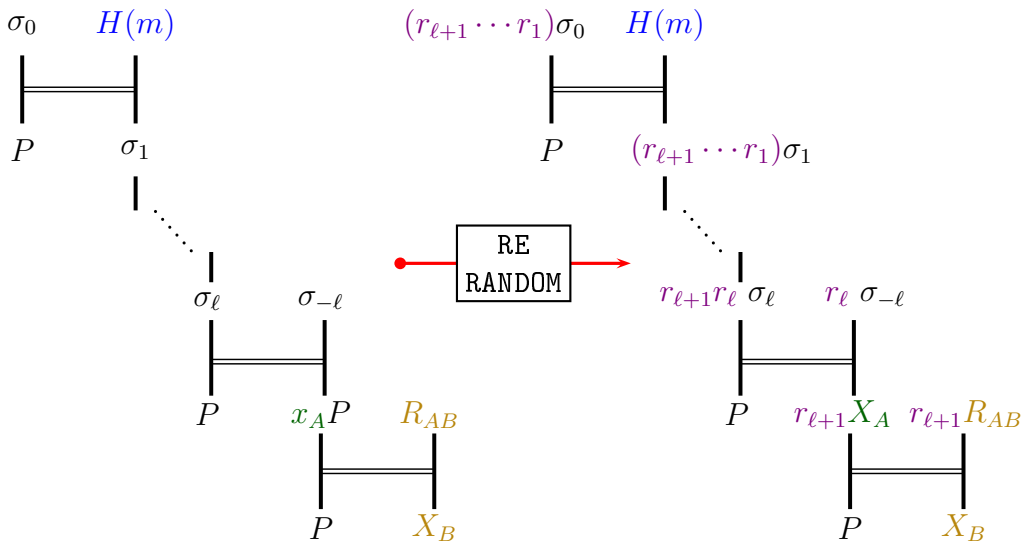


Figure 9.8: Step 2: Re-randomization of the re-signature

10. Formal definition

First we recall the formal definition of a *proxy re-signature* scheme from Ateniese & Hohenberger (2005).

DEFINITION 10.1. *A (uni-directional) proxy re-signature scheme for N signers and L levels consists of the following six randomized algorithms:*

1. ***Setup*(λ):** *On input of the security parameter λ, this randomized algorithm produces the public system parameters* **cp**. *It will be run by a trusted party.*

2. ***KeyGen*(cp):** *On input of the public parameters* **cp** *this probabilistic algorithm outputs a users' public and private key pair (pk, sk).*

3. ***ReKeyGen*(cp, pk_i, sk_j):** *On input of the public parameters* **cp**, *signer i's public key pk_i and user j's private key sk_j this (non-interactive) algorithm outputs a re-signature key R_{ij} that allows translating signatures of user i into signatures of user j.*

4. ***Verify*(cp, $m, \ell, \sigma_i^{(\ell)}, pk_i$):** *On input of the public parameters* **cp**, *a message $m \in \{0,1\}^*$, a level ℓ signature $\sigma^{(\ell)}$ and a public key pk_i, this de-*

terministic algorithm outputs 1 *if* $\sigma^{(\ell)}$ *is a valid signature for* pk_i *or* 0 *if otherwise.*

5. ***Sign(*****cp**$, m, \ell, sk_i$***):*** *On input of the public parameters* **cp**, *a message* $m \in \{0,1\}^*$ *and the signers private key* sk_i *this randomized algorithm outputs a signature* $\sigma_i^{(\ell)}(m)$ *of user* i *on the message* m *at level* $\ell \in \{0, ..., L\}$.

6. ***Re-Sign(*****cp**$, m, \ell, \sigma_i^{(\ell)}, R_{ij}, pk_i, pk_j$***):*** *Given the public parameters* **cp**, *a message* $m \in \{0,1\}^*$, *a level* ℓ *signature* $\sigma_i^{(\ell)}$ *of user* i, *the re-signature key* R_{ij}, *this randomized algorithm produces* $\sigma_j^{(\ell+1)}$ *a level* $\ell+1$ *signature for user* j *if* $\sigma_i^{(\ell)}$ *is valid for the public key* pk_i.

Here λ *is the security parameter and both the number* N *of users and the number* L *of allowed translations (levels) are polynomial in* λ. *We require that for all* $\lambda \in \mathbb{N}$ *all system parameters* **cp** *produced by* ***Setup()***, *for all public and private key-pairs* $(pk_i, sk_i), (pk_j, sk_j)$ *produced by* ***KeyGen()*** *and for any* $\ell \in \{0, ..., L\}$ *and messages* $m \in \{0,1\}^*$:

$$\mathbf{Verify}\,(\mathbf{cp}, m, \ell, \mathbf{Sign}(\mathbf{cp}, m, \ell, sk_i), pk_i) = 1\ .$$

$$\mathbf{Verify}(\mathbf{cp}, m, \ell, \mathbf{ReSign}(\mathbf{cp}, m, \ell, \mathbf{Sign}(\mathbf{cp}, m, \ell, sk_i), \mathbf{ReKeyGen}(\mathbf{cp}, pk_i, sk_j)), pk_j) = 1\ .$$

Now we will specify the implementation of the *multi-use uni-directional proxy re-signature scheme* from Libert & Vergnaud (2008a).

The multi-use uni-directional scheme

Setup(λ**):** On input of the security parameter $\lambda \in \mathbb{N}$, this algorithm chooses bilinear groups $(\mathbf{G}, \mathbf{G}_T)$ of prime order $p > 2^\lambda$, a generator $P \in \mathbf{G}$ and a hash function $H\colon \{0,1\}^* \to \mathbf{G}$. The public system parameters are

$$\mathbf{cp} := \{\mathbf{G}, \mathbf{G}_T, P, H\}.$$

Keygen(cp): User i's public and private key pair is $(X_i,\ x_i)$ with a random $x_i \xleftarrow{R} \mathbb{Z}_p^\times$.

ReKeygen($\mathbf{cp}, X_i, x_j$**):** This algorithm outputs the Re-Signature key as $R_{ij} = \frac{1}{x_j} X_i = \frac{x_i}{x_j} P$ which allows the proxy to translate signatures of user i into signatures of user j.

Verify(cp, $m, \ell, \sigma^{(\ell)}, X_i$**):** The validity of an level ℓ signature

$$\sigma^{(\ell)}(m) = (\sigma_0, \sigma_1, \ldots, \sigma_\ell, \sigma_{-\ell}, \ldots, \sigma_{-1}) \in \mathbf{G}^{2\ell+1}$$

on a message $m \in \{0,1\}^*$ for the public key X_i is verified if the $\ell+1$ equations

$$e(\sigma_0, P) \stackrel{?}{=} e(\sigma_1, H(m)), \tag{10.2}$$

$$e(\sigma_k, P) \stackrel{?}{=} e(\sigma_{k+1}, \sigma_{-k}) \text{ for } k \in \{1, \ldots, \ell-1\}, \tag{10.3}$$

$$e(\sigma_\ell, P) \stackrel{?}{=} e(X_i, \sigma_{-\ell}) \tag{10.4}$$

all hold. For $\ell = 0$ this specializes to the following:

$$e(\sigma^{(0)}, P) \stackrel{?}{=} e(X_i, H(m)).$$

Sign(cp, m, ℓ, x_i**):** On input a message $m \in \{0,1\}^*$ and a private key x_i, this algorithm signs signatures at level ℓ for user i. It first computes

$$\sigma^{(0)}(m) = x_i H(m).$$

Then appends ℓ trivial H-s and re-randomizes the result. During the computation the algorithm chooses $r_1, \ldots, r_\ell \xleftarrow{\$} \mathbb{Z}_p^\times$ and computes and outputs the group elements as

$$\sigma_0^{(\ell)} = r_\ell \cdot \cdots \cdot r_2 \cdot r_1 \cdot x_i H(m),$$

$$\sigma_k^{(\ell)} = r_\ell \cdots r_k \cdot x_i P \qquad \text{for } k \in \{1, \ldots, \ell\},$$

$$\sigma_{-k}^{(\ell)} = r_k \cdot P \qquad \text{for } k \in \{\ell, \ldots, 1\}.$$

Note that this is precisely as in Figure 8.5.

Re-Sign(cp, $m, \ell-1, \sigma^{(\ell-1)}, R_{ij}, X_i, X_j$**):** On input a message $m \in \{0,1\}^*$, a *valid* level $\ell-1$ signature

$$\sigma^{(\ell-1)}(m) = (\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_{\ell-1}, \sigma_{-\ell+1}, \ldots, \sigma_{-1}) \in \mathbf{G}^{2\ell-1}$$

on m, the re-signature key $R_{ij} = \frac{x_i}{x_j} P$ and the public keys X_i, X_j, this algorithm re-sings $\sigma^{(\ell-1)}$ to $\sigma^{(\ell)}$ valid for X_j. It first appends the re-signature H and then re-randomizes the result. During the computation the algorithm chooses ℓ random elements $t_1, t_2, \ldots, t_\ell \xleftarrow{\$} \mathbb{Z}_p^\times$ then it translates $\sigma^{(\ell-1)}$ into a level ℓ

signature valid for X_j by computing and outputting

$$\sigma_0^{(\ell)} = t_\ell \ldots\ldots t_2 t_1 \cdot \sigma_0^{(\ell-1)},$$

$$\sigma_k^{(\ell)} = t_\ell \ldots t_k \cdot \sigma_k^{(\ell-1)} \qquad \text{for } k \in \{1, \ldots, \ell-1\},$$

$$\sigma_\ell^{(\ell)} = t_\ell X_i,$$

$$\sigma_{-\ell}^{(\ell)} = t_\ell R_{ij} = t_\ell \tfrac{x_i}{x_j} P,$$

$$\sigma_{-k}^{(\ell)} = t_k \cdot \sigma_k^{(\ell-1)} \qquad \text{for } k \in \{\ell-1, \ldots, 1\}.$$

We observe that:

$$\begin{aligned}\sigma^{(\ell)}(m) &= (\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_\ell, \sigma_{-\ell}, \ldots, \sigma_{-2}, \sigma_{-1}) \\ &= (\overline{r}_\ell \ldots\ldots \overline{r}_1 x_j H(m), \\ &\quad \overline{r}_\ell \ldots\ldots \overline{r}_1 x_j P, \\ &\quad \overline{r}_\ell \ldots \overline{r}_2 x_j P, \\ &\quad \vdots \\ &\quad \overline{r}_\ell x_j P, \\ &\quad \overline{r}_\ell P, \ldots, \overline{r}_2 P, \overline{r}_1 P) \in \mathbf{G}^{2\ell+1}.\end{aligned}$$

If we set $\overline{r}_\ell = t_\ell \frac{x_i}{x_j}$ and $\overline{r}_k = t_k r_k$ for $k \in \{1, \ldots, \ell-1\}$. Since

$$\begin{aligned}\sigma^{(\ell-1)}(m) &= (\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_{\ell-1}, \sigma_{-\ell+1}, \ldots, \sigma_{-1}) \\ &= (\ r_{\ell-1} \cdot \cdots \cdot r_1 x_i H(m), \\ &\quad r_{\ell-1} \cdot \cdots \cdot r_1 x_i P, \\ &\quad r_{\ell-1} \cdots r_2 x_i P, \\ &\quad \vdots \\ &\quad r_{\ell-1} x_i P, \\ &\quad r_{\ell-1} P, \ldots, r_2 P, r_1 P) \in \mathbf{G}^{2\ell-1}.\end{aligned}$$

Notice the slight change of the order of the elements in (10.2) and (10.4) in the verification process. This notion is formally more correct because we can use two different groups $\mathbf{G}_1$ and $\mathbf{G}_2$ instead of one to construct the signature scheme. Recall the H representation of a level ℓ signature as in Figure 8.4, in each H we then would have the diagonal elements in the same groups. This is called the *asymmetric* case and the choice of using two groups instead of one is highly related to the security of the signature scheme which we discuss in the next chapter. We also observe the following:

- *Uni-directional:* The re-signature key R_{ij} allows the proxy to translate signatures in one direction.
- *Multi-use:* Signatures in this scheme can be translated polynomially many times, in fact a signer can limit the number of translations to n by signing the signature at level $L - n$ where at most L translations are allowed.
- *Private-proxy:* An honest proxy can keep the re-signature key R_{ij} secret because while translating the signature, R_{ij} is blinded out by a random element $t_\ell \xleftarrow{\$} \mathbb{Z}_p^\times$.
- *Transparency:* Since the signatures can also be signed at some arbitrary level $\ell \in \{0, \ldots, L\}$ the user does not even know that a proxy exists.
- *Unlinkability:* A signature translated from level $\ell - 1$ has the same distribution of the coefficients as a signature which was signed at level ℓ, thus a user has no way of linking it to its predecessor.
- *Non-interactive:* Trivially, the re-signature key $R_{ij} = \frac{x_i}{x_j} P$ can be calculated without the interaction of the delegatee i. As mentioned before signer j can calculate R_{ij} by $\frac{1}{x_j} X_i$ and make it available to the proxy for example with an interactive secure protocol.
- *Non-transitive:* The proxy cannot re-delegate his signing rights. This means that even if he is in possession of R_{AB} and R_{BC} he is not able to produce R_{AC} for some users `A`,`B` and `C`. Note that in this case the proxy can first translate the signatures of `A` into ones of `B` and then translate them into signatures of `C` but he is not able to translate signatures directly.
- *Key-optimal:* Signers only have to store a constant amount of data, ie. one private key for `Aylin` is enough for all signatures and delegations she makes.

Part III
Security

In this chapter we will discuss the security issues of the signature scheme. We will start with reviewing the underlying cryptographic assumptions and after that we formulate an adversary model and compare the two different environments in which the adversaries are simulated. We will continue with formulating a *new* security definition and compare it the original security definition defined in Ateniese & Hohenberger (2005) and also used in Libert & Vergnaud (2008a).

Using this new security definition we first prove that the signature scheme is secure in the random oracle model (Bellare & Rogaway 1993). Then we modify the signature scheme with a trick from Waters (2005) and prove that the signature scheme is also secure in the standard model after this slight modification.

11. Cryptographic assumptions

We first recall the definition of a bilinear pairing from Part I.

DEFINITION 11.1. *For prime order groups $\boldsymbol{G}$ and $\boldsymbol{G}_T$ a bilinear map $e\colon \boldsymbol{G} \times \boldsymbol{G} \to \boldsymbol{G}_T$ is a mapping with the following properties*

(i) *e is bilinear: $e(aP, bQ) = e(P, Q)^{ab}$ for all $(P, Q) \in \boldsymbol{G} \times \boldsymbol{G}$ and $a, b \in \mathbb{Z}_p^\times$.*

(ii) *e is non-degenerate: $e(P, P) \neq 1$ for some $P \in \boldsymbol{G}$.*

(iii) *e is efficiently computable.*

The symmetric setting Although we defined pairings in Part I generally in an *asymmetric* setting, here we use the *symmetric* setting for the signature scheme as we did in Part II. In practice the security of the signature scheme is highly related to the embedding degree k of the elliptic curve $E(\mathbb{F}_{q^k})$ on which the target group $\mathbf{G}_T$ is defined. The security level β is measured in bits which means that calculating the relevant discrete logarithm should take approximately 2^β basic operations. The National Institute of Standards and Technology (NIST) recommends until the end of year 2010 an 80 bit security level (Barker, Barker, Burr, Polk & Smid 2007). This implies that in RSA based cryptosystems the key size has to be at least 1024 bits (after 2010 even 2048

bits) which corresponds approximately to a key size of at least 160 bits (after 2010 at least 224 bits) in elliptic curve based systems (Barker, Burr, Jones, Polk, Rose, Dang & Smid 2009). Thus, with an 80 bit security requirement in the symmetric setting we need the prime order q of the group $\mathbf{G}$ at least to be 160 bits, ie. $q \approx 2^{160}$. Now consider the target group $\mathbf{G}_T$. Since this is a multiplicative subgroup of $\mathbb{F}_{q^k}$ we need 1024 bits to achieve 80 bit security there. Therefore to achieve 80 bits of security we need $q^k \approx 2^{1024}$. This means that to meet the recommendation of NIST for an 80 bit security level we need at least $k = 6$. Unfortunately in the symmetric setting, for $k = 6$ there are not many curves available. And if we try to work the opposite way around for a smaller embedding degree, let's say $k = 2$, we end up with inefficient groups of size $q \approx 2^{512}$ which is much higher than the required value 2^{160}. Consequently the situation gets worse for 128 bit and 256 bit security levels, but we can modify our signature scheme into an *asymmetric* setting in which there are more curves available for $k \geq 6$. A good family of curves can be found in Barreto & Naehrig (2005). We can modify our signature scheme into an asymmetric setting by allowing two groups $\mathbf{G}_1$ and $\mathbf{G}_2$. The elements of a level ℓ signature are then distributed as follows:

$$\begin{aligned}
&\sigma_i^{(\ell)} \in \mathbf{G}_1 \text{ for all } i \in \{0, \ldots, \ell\},\\
&\sigma_{-i}^{(\ell)} \in \mathbf{G}_2 \text{ for all } i \in \{1, \ldots, \ell\} \text{ and}\\
&H(m), P \in \mathbf{G}_2.
\end{aligned}$$

Consider Figure 11.1, in each H the diagonal elements must be from the same group. However, to simplify things for the theoretical approach we only consider the *symmetric* setting throughout this chapter.

The signature scheme relies on the generic Computational Diffie-Hellman (CDH) assumption and on another current generalization of it, called the ℓ-flexible Diffie-Hellman (ℓ-flexDH) problem. Recalling the definition of the well known generic computational Diffie-Hellman problem we introduce the ℓ-flexDH problem. We assume that the discrete logarithm problem is hard in these groups.

DEFINITION 11.2. *The* ***computational Diffie-Hellman*** *(CDH) problem is, given $P, aP, bP \in \boldsymbol{G}$ to compute $abP \in \boldsymbol{G}$. The computational Diffie-Hellman assumption is that this problem is hard to solve.*

To introduce and understand the ℓ-flexDH problem, we start with the definition of the 1-flexDH problem.

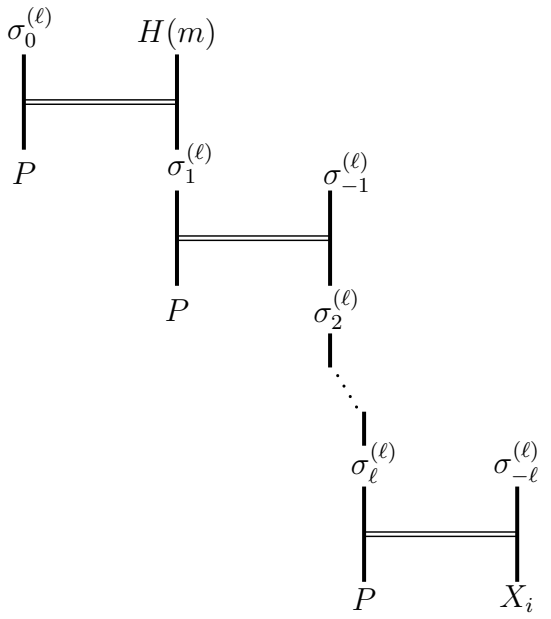


Figure 11.1: A level ℓ signature $\sigma^{(\ell)}$ valid for X_i

DEFINITION 11.3. *The* ***1-flexible Diffie-Hellman*** *(1-flexDH) problem is, given* $P, aP, bP \in \boldsymbol{G}$ *to compute a triple* $(abC, aC, C) \in \boldsymbol{G}^3$ *such that* C *is not the neutral element of the group. The 1-flexible Diffie-Hellman assumption is that this problem is hard to solve.*

The 1-flexDH problem is very similar to what is known as 2-out-of-3 Diffie-Hellman problem which states that, it is already hard to compute a pair $(abC, C) \in \mathbf{G}^2$ from the same triple $(P, aP, bP) \in \mathbf{G}^3$. Now we extend the definition of the 1-flexDH problem to the ℓ-flexDH.

DEFINITION 11.4. *The* ℓ***-flexible Diffie-Hellman*** *(ℓ-flexDH) problem is, given* $P, aP, bP \in \boldsymbol{G}$ *to find a* $(2\ell+1)$*-tuple*

$$(abD_\ell, aD_\ell, \ldots, aD_1, C_\ell, \ldots, C_1) \in \boldsymbol{G}^{2\ell+1}$$

such that $\log_P D_j = \prod_{i=1}^{j} \log_P C_i$ *for all* $i \in \{1, \ldots, \ell\}$*, where* C_i *is not the neutral element of the group* $\mathbf{G}$*.*

THEOREM 11.5. *The ℓ-flexDH problem as defined above is hard to solve.*

PROOF. A proof of hardness of the ℓ-flexDH in generic groups can be found in Libert & Vergnaud (2008a). The proof uses the family of computational problems provided in Kunz-Jacques & Pointcheval (2006) which allow the study of the variants of the CDH in the generic group model. □

For explanatory purposes we also introduce another variant of the CDH which is called the *modified computational Diffie-Hellman* (mCDH) problem.

DEFINITION 11.6. *The* ***modified computational Diffie-Hellman*** *(mCDH) problem is, given $P, aP, \frac{1}{a}P, bP \in \boldsymbol{G}$ to compute $abP \in \boldsymbol{G}$. The modified computational Diffie-Hellman assumption is that this problem is hard to solve.*

12. Adversary model

We now pick up the security discussion which we skipped at the end of Part II. The general security notion for signatures considers an adversary against the scheme and addresses two issues:

1. the capabilities of the adversary, in particular, how much information does the adversary have,

2. and the adversary goal.

We consider the security of the signature scheme as *existentially unforgeable* against an adversary with *adaptive chosen message attack* capabilities (EUF-CMA). This means that we consider an adversary $\mathcal{A}$ with full access to the signer who is idealized as a signing oracle and to the public key of the signer. More concretely, $\mathcal{A}$ is allowed to query the signing oracle to obtain valid signatures $\sigma_1, \ldots, \sigma_n$ on arbitrary messages $m_1, \ldots, m_n$. Since $\mathcal{A}$ can adaptively ask for signatures on different messages this is called an *adaptive chosen message attack*. As an example for an adversary with a lot less capabilities, we could limit the adversary to have no information at all except the public key. This would result in an *key only attack* (KOA) which is the unavoidable case anyway.

In the end $\mathcal{A}$ is considered to be successful if he can come up with a signature σ^* on a message $m^* \notin \{m_1, \ldots, m_n\}$ within reasonable time. Notice that $\mathcal{A}$ is required create a new message m^* and a new signature σ^* by himself and this is called *existentially unforgeability* of the signature scheme. This is a very strong

security requirement since the goal of the adversary is very "easy". This is because we require $\mathcal{A}$ to calculate only one forgery. As an example of a weaker security requirement than EUF, thus a harder goal for $\mathcal{A}$, we could require $\mathcal{A}$ to produce a forgery σ^* for a given message m^*. This is called *selective unforgeability* (SUF). The weakest security requirement and thus the hardest goal for an adversary in this sense is that we require him to recover the secret key sk from the given public key pk. This is called *unbreakability* (UB).

12.1. Strong unforgeablity. A slightly stronger security requirement than EUF is the *strong unforgeability* (SEUF) as defined in An, Dodis & Rabin (2002). Here we require the attacker not only be unable to forge a signature of a "new" message, but also that he is unable to generate even a different signature from an already signed message, ie. we only require $\sigma^* \notin \{\sigma_1, \ldots, \sigma_n\}$. In this sense we observe that the signature scheme considered in this thesis is not SEUF, since any level $\ell \geq 1$ signature can be publicly re-randomized. It seems that in this setting we cannot have SEUF if we want *unlinkability*. This results from the fact that the unlinkability is achieved through the re-randomization step in the re-singing process. Therefore, this is not a weakness of the signature scheme, on the contrary it allows the desired *unlinkability* property.

12.2. The adversary. Translated into daily language EUF-CMA security means that, even if an attacker `Charly` has access to his victim `Aylin`'s computer for a while and produces many valid signatures of her on arbitrary messages of his choice, he is still unable to produce a valid message signature pair (m^*, σ^*) on a fresh message m^* by himself. In SEUF-CMA security, `Charly` would even be unable to produce a new message signature pair (m^*, σ^*) from the signatures he obtained from `Aylin`'s computer. The difference is that in EUF-CMA security we do not allow `Charly` to query the message of m^* at any time. In the SEUF-CMA security however, `Charly` is allowed to query a signature for m^* but is required to create a new signature for m^* in that case.

In Goldwasser, Micali & Rivest (1988) the highest security level is considered as EUF-CMA. This means that an adversary who cannot produce an existential forgery with adaptive chosen message attack capabilities is also not able to forge signatures on weaker security notions with lesser capabilities, for example SUF-KOA. Thus it is desirable to prove the security with respect to EUF-CMA. To formalize this we define the following game

EUF-CMA Game.

Input: An attacker $\mathcal{A}$, a signature oracle $\mathcal{O}_{\text{Sign}}$, the list (pk) of public keys of all possible victims.

Output: { WIN , LOOSE }.

1. $(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}\,((pk), \mathcal{O}_{\text{Sign}})$.
2. If σ^* is valid for pk^* on m^* then
3. If m^* was queried from $\mathcal{O}_{\text{Sign}}$ then
   LOOSE .
4. Else WIN .
5. Else LOOSE .

The adversary $\mathcal{A}$ has access to the public keys pk of all possible victims and a signature oracle $\mathcal{O}_{\text{Sign}}$ which returns signatures on behalf of the victims to $\mathcal{A}$. After a while $\mathcal{A}$ outputs a message signature pair (m^*, σ^*) valid for the public key pk^* (step 1). Obviously we do not allow $\mathcal{A}$ to ask $\mathcal{O}_{\text{Sign}}$ for a signature on m^* (step 3). We require that $\mathcal{A}$ has at most advantage ε of winning the game, ie.

$$Pr[\mathcal{A} \text{ wins EUF-CMA Game}] \leq \varepsilon.$$

A proof of security states that if there exists an attacker $\mathcal{A}$ who can break EUF-CMA security then this also implies breaking the underlying cryptographic assumptions which in our case is the ℓ-flexDH assumption. Such a proof is actually a reduction from the attacker $\mathcal{A}$ to the underlying assumption, ie. if an attacker $\mathcal{A}$ exists we can use it as a blackbox and solve the cryptographic problem by manipulating $\mathcal{A}$'s input and oracles.

In the following sections we will show that the signature scheme is secure by constructing algorithms which use EUF-CMA attackers with advantage ε to solve the ℓ-flexDH instances. We will also allow these attackers to have access to more information through different oracles. There are two different environments (models) used in security proofs for simulating the attackers while manipulating their inputs and oracles. These are:

1. the random oracle model,
2. and the standard model.

Each model has its advantages and disadvantages which we will briefly discuss now.

12.3. Random oracle model versus standard model. The random oracle model was introduced in Bellare & Rogaway (1993) and has been a useful tool for proving the security of many signature schemes ever since (over 2400 citations on Google scholar). In the random oracle model the attacker has another oracle $\mathcal{O}_{\text{Hash}}$ which he can query for hash values on arbitrary messages of

his choice. This idealization of the hash function as an oracle in this environment results from the assumption that the hash function used in the signature scheme is truly random and consequently that the attacker is independent of the hash function. However, when moving from the theoretical scheme to a practical implementation, the idealized hash function has to be implemented as a certain cryptographic function such as SHA-1 or MD5 (Barker *et al.* 2009; Eastlake & Jones 2001; Rivest 1992). This means that there are signatures schemes which are secure in the random oracle model but have no secure implementation in a real world without random oracles (Canetti, Goldreich & Halevi 2004). The weakness of the random oracle model is that the attacker can also be dependent of the hash function, such that he is exploiting specific flaws in the actual implementation of the hashing oracle. This would mean that the modifications made to the hashing oracle would also corrupt the output of the attacker which would also make the reduction invalid in this model.

In the standard model there are no idealizations except the signing oracle which is realistic as described above. More importantly this makes the scheme only stronger. This means that, in this environment, the hash function has a certain implementation and therefore a certain probability distribution of its outputs. When manipulating the hash function we have to assure that after the manipulations the distribution of the outputs is still the same. In contrast to the random oracle model this complicates a reduction in the standard model. In fact later we even modify the actual signature scheme slightly to achieve provable security in the standard model. Thus, even though the random oracle model is a powerful tool for the theoretical approach it is still desirable to have a security proof in the standard model. For more information about the capabilities and a detailed discussion of the shortcomings of the random oracle model we refer to the link farm Lipmaa (2010).

13. Security definition

In this section we provide a new security definition for the signature scheme considering a generic adversary $\mathcal{A}$ who has access to as much as information possible. We allow $\mathcal{A}$ to have access to four different oracles and keep track of $\mathcal{A}$'s oracle calls in a query list Q-list. When $\mathcal{A}$ outputs a forgery we construct a directed graph from the query list. The idea is to rule out the combination of $\mathcal{A}$'s queries which lead to a trivial forgery.

13.1. The oracles. There are four different oracles available to $\mathcal{A}$.

1. $\mathcal{O}_{\text{Skey}}(i)$. When queried with i, the private key oracle returns the private

key sk_i of user i. The entry $[\mathcal{O}_{\text{Skey}}, i]$ is added to Q-list.

2. $\mathcal{O}_{\text{ReKey}}(i, j)$. When queried with (i, j) the re-signature key oracle returns R_{ij} the re-signature key from user i to j. The entry $[\mathcal{O}_{\text{ReKey}}, (i, j)]$ is added to Q-list.

3. $\mathcal{O}_{\text{Sign}}(i, m, \ell)$. When queried with (i, m, ℓ), the signature oracle returns a level ℓ signature σ_i on m valid for pk_i the public key of user i. The entry $[\mathcal{O}_{\text{Sign}}, (i, m)]$ is added to Q-list.

4. $\mathcal{O}_{\text{ReSign}}(\sigma_i, i, j, m)$. When queried with (σ_i, i, j, m), the re-signature oracle returns σ_j a re-signature of σ_i on m valid for pk_j the public key of user j. The entry $[\mathcal{O}_{\text{ReSign}}, (i, j, m)]$ is added to Q-list.

When $\mathcal{A}$ outputs a forgery, the entries in Q-list consists of tuples [oracle, query].

13.2. The query graph. The query graph G_Q is constructed from the query list after $\mathcal{A}$ comes up with a forged message signature pair (m^*, σ^*) valid for pk_{i^*} the public key of user i^*. Consider the following algorithm:

QUERY-GRAPH.

Input: A user i^*, a message m^* and a query list Q-list,
Output: A directed graph $G_Q = (V, E)$.

1. $M := \{m \mid \exists\, i, j : [\mathcal{O}_{\text{Sign}}, (i, m)] \in Q\text{-list or } [\mathcal{O}_{\text{ReSign}}, (i, j, m)] \in Q\text{-list}\}$.
2. $V \leftarrow \{[0], [i^*, m^*]\}$, $E \leftarrow \emptyset$.
3. For each entry $[\text{oracle}, \text{query}] \in Q$-list Do 3-15.
4. If oracle $= \mathcal{O}_{\text{Skey}}$ && query $= i$ then
5. $V \leftarrow V \cup \{[i, m]\}$ for all $m \in M$.
6. $E \leftarrow E \cup \{([0], [i, m])\}$ for all $m \in M$.
7. Else if oracle $= \mathcal{O}_{\text{ReKey}}$ && query $= (i, j)$ then
8. $V \leftarrow V \cup \{[i, m], [j, m]\}$ for all $m \in M$.
9. $E \leftarrow E \cup \{([i, m], [j, m])\}$ for all $m \in M$.
10. Else if oracle $= \mathcal{O}_{\text{Sign}}$ && query $= (i, m)$ then
11. $V \leftarrow V \cup \{[i, m]\}$.
12. $E \leftarrow E \cup \{([0], [i, m])\}$.
13. Else if oracle $= \mathcal{O}_{\text{ReSign}}$ && query $= (i, j, m)$ then
14. $V \leftarrow V \cup \{[i, m], [j, m]\}$.
15. $E \leftarrow E \cup \{([i, m], [j, m])\}$.
16. Return $G_Q \leftarrow (V, E)$

In the beginning the algorithm defines a set M containing all queried messages m from the Q-list (step 1). The algorithm then initializes the set of nodes V with a start node $[0]$ and the final node $[i^*, m^*]$, at this point the set of edges E is empty (step 2). Each private key sk_i queried from $\mathcal{O}_{\text{Skey}}$, allows an attacker $\mathcal{A}$ to create signatures on behalf of user i on any message $m \in M$, especially on $m^* \in M$. Therefore the nodes labeled $[i, m]$ with edges $([0], [i, m])$ for all $m \in M$ are added to the graph (steps 5 and 6). Similarly each re-signature key R_{ij} allows an attacker to translate a signature σ_i of user i into a signature σ_j of user j independent of the signed message m. Thus, the algorithm adds the nodes $[i, m]$, $[j, m]$ and the interconnecting edges $([i, m], [j, m])$ for all $m \in M$ to the graph (steps 8 and 9). For each signature query of user i on m the node $[i, m]$ and the edge $([0], [i, m])$ are added to the graph (steps 11 and 12). Also for each re-signature query from user i to user j on a message m the node $([i, m])$, $([j, m])$ with the interconnecting edge $([i, m], [j, m])$ are added to the graph (steps 14 and 15). In the end if there is a path from $[0]$ to $[i^*, m^*]$ we know for sure that $\mathcal{A}$ has obtained some information which allows him to create (m^*, σ^*) trivially.

Note that in view of generality we could also label the nodes additionally with the queried signatures ie. $[i, m, \sigma]$. However this does not make sense in our case because the signature scheme is not *strongly unforgeable* (SEUF). This means that even if we use such a labeling of nodes, we are not interested in the additional information σ since the adversary is able to transform σ into σ' by himself. Thus considering only the queried users i and the queried messages m is enough for us to know if $\mathcal{A}$ has some information which leads to a trivial forgery. Further, we could also only treat the queries containing m^* since in the end we are only interested in a path from $[0]$ to $[i^*, m^*]$.

13.3. The challenge. Now we define the following game:

GAME 13.1.

Public input: A list (pk_i) of public keys of all users $i \in \{0, \ldots, N-1\}$.

Input: An attacker $\mathcal{A}$, a private key oracle $\mathcal{O}_{\text{SKey}}$, a signature oracle $\mathcal{O}_{\text{Sign}}$, a re-signature oracle $\mathcal{O}_{\text{ReSign}}$ and a re-signature key oracle $\mathcal{O}_{\text{ReKey}}$.

Output: { WIN , LOOSE }.

1. $(i^*, m^*, \sigma^*) \leftarrow \mathcal{A}\,((pk_i), \mathcal{O}_{\text{SKey}}, \mathcal{O}_{\text{ReKey}}, \mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{ReSign}})$.
2. If $\text{Verify}(\cdot, pk_{i^*}, m^*, \sigma^*) = 0$ then LOOSE .
3. $G_Q \leftarrow$ QUERY-GRAPH$(i^*, m^*, Q\text{-list})$.
4. If there is a path from $[0]$ to $[i^*, m^*]$ in G_Q then LOOSE .
5. Else WIN .

The attacker $\mathcal{A}$ is allowed to have access to all the public keys (pk_i) of users $i \in \{0, \ldots, N-1\}$ and all the oracles defined above (step 1). $\mathcal{A}$ is then required to come up with a forged message signature pair (m^*, σ^*) valid for pk_{i^*} the public key of user $i^* \in \{0, \ldots, N-1\}$ (steps 1 and 2). Then the query graph G_Q is constructed from the Q-list with respect to i^* and m^* (step 3). If there is no path from $[0]$ to $[i^*, m^*]$ in G_Q, $\mathcal{A}$ wins Game 13.1.

13.4. The new security definition. We call the signature scheme secure if for any attacker $\mathcal{A}$ the probability $Pr[\mathcal{A}\text{ wins Game 13.1}] = f(\lambda)$ is negligible in the security parameter λ. Recall that f is negligible in λ iff $\forall p \in \text{poly}^+(\lambda)\ \exists L\ \forall \lambda > L :\ f(\lambda) \leq \frac{1}{p(\lambda)}$, where $\text{poly}^+(\lambda)$ denotes the set of positive polynomials in λ.

In our new security definition we consider a generic attacker $\mathcal{A}$ who has access to four different oracles. We keep track of $\mathcal{A}$'s oracle queries in a query list Q-list where the oracle calls and matching query values are recorded together as entries in the form of $[\text{oracle}, \text{query}]$. The graph G_Q generated by the algorithm Query-graph from Q-list allows us to determine if $\mathcal{A}$ queried some information which leads to a trivial forgery, ie. if $\mathcal{A}$ cheated. This means that $\mathcal{A}$ is not restricted in anyway and has access to as much as information possible.

Now we recall and compare the old security definition from Ateniese & Hohenberger (2005) which was also used in Libert & Vergnaud (2008a) with our new security definition.

13.5. The security definition from Ateniese & Hohenberger (2005). This security definition for uni-directional proxy re-signature schemes distinguishes between internal and external security.

13.5.1. Internal security. This notion captures that an honest party inside the system is secure against colliding delegation partners. There are three different security notions defined inside the system depending on which delegation partner an attacker, ie. a corrupt user, can impersonate.

Limited proxy security. This notion captures the inability of the proxy to sign messages on behalf of the delegatee (who's signature is translated) and also his inability to create signatures for the delegator unless they were first signed by the delegatee. In this definition the adversary is allowed to have access to all public keys, all re-signature keys and a signature oracle which returns level 0 signatures on behalf of *any* user. In the end the adversary is required to come up with any level ℓ signature σ^* valid for some user $i^* \in \{0, \ldots, N-1\}$ of his choice. The adversary fails if he queried the signature oracle for a signature on

the forged message m^* before. The proof uses this adversary with advantage ε to solve a given ℓ-flexDH instance with a success probability $\Omega(\frac{\varepsilon}{q_s})$ where q_s is the number of signature queries.

In comparison to our new security definition and the other adversaries in this security definition this adversary seems to be the most "natural" one. We note that this adversary $\mathcal{A}_1$ has access to all re-signature keys and therefore can create re-signatures on his own. We also note that $\mathcal{A}_1$ has not access to any secret key information. Thus, we can reduce $\mathcal{A}_1$ to our generic adversary $\mathcal{A}$ by limiting $\mathcal{A}$'s access to the oracles $\mathcal{O}_{\text{ReKey}}$, $\mathcal{O}_{\text{Sign}}$ and $\mathcal{O}_{\text{ReSign}}$.

Delegatee security. This notion states that an honest delegatee (who's signature is translated) is protected against a colliding delegator and proxy. This means that an attacker impersonating as the proxy and the delegator has very little chance of coming up with a forgery on behalf of the targeted delegatee. In this definition the adversary is allowed to have access to all public keys, all secret keys except the secret key of the targeted delegatee i^*, all re-signature keys except R_{ii^*}, and a signature oracle which returns level 0 signatures on behalf of targeted delegatee. In the end the adversary is required to come up with a level ℓ signature σ^* on behalf of the targeted delegatee i^*. The adversary fails if he queried the signature oracle for a signature on the forged message m^* before. The proof uses this adversary with advantage ε to solve a given ℓ-flexDH instance with a success probability $\Omega(\frac{\varepsilon}{q_s})$ where q_s is the number of signature queries.

This adversary $\mathcal{A}_2$ is not allowed to choose freely which user he wants to corrupt, in comparison to our generic adversary $\mathcal{A}$ this seems like a very unnatural limitation of the security notion EUF-CMA. We can reduce $\mathcal{A}_2$ to $\mathcal{A}$ by requiring $\mathcal{A}$ to come up with a signature σ^* valid for the public key pk_{i^*} of a user i^* who is fixed in the beginning. The access to all secret keys sk_i of users $i \in \{0, \ldots, N-1\} \setminus \{i^*\}$ allows $\mathcal{A}_2$ to create signatures for users $i \neq i^*$ by himself. Further the access to all re-signature keys R_{ij} where $j \neq i^*$ allows him also to create re-signatures of his choice. Note that in the new security definition the query for the secret key sk_{i^*} would create a path from $[0]$ to $[i^*, m^*]$ in the query graph G_Q. However, the query for the re-signature key R_{ii^*} would create an edge $([i, m^*], [i^*, m^*])$ but not necessarily a path from $[0]$ to $[i^*, m^*]$ because the simple knowledge of R_{ii^*} does not lead to a trivial forgery. This means that in the old security definition this permissible query is ruled out and not considered because the adversary $\mathcal{A}_2$ has access to *all* secret keys sk_i except sk_{i^*}. Thus, the difference between $\mathcal{A}$ and $\mathcal{A}_2$ is not only the free choice of targeted delegatee i^* but also the allowed queries for re-signature keys R_{ii^*}

which do not necessarily lead to a trivial forgery.

Delegator security. This notion captures that a collision between the delegatee and the proxy is harmless for an honest delegator. This means that the attacker who is impersonating the delegatee and the proxy has very little chance of coming up with a forgery on behalf of the targeted delegator. In this definition the adversary is allowed to have access to all public keys, all secret keys except the secret key of the targeted delegator i^*, all re-signature keys and a signature oracle which returns level 0 signatures on behalf of target delegatee. In the end the the adversary is required to come up with a level 0 signature σ^* for the targeted delegator i^*. The adversary fails if he queried the signature oracle for a signature on the forged message m^* before. The proof uses this adversary with advantage ε to solve the given mCDH instance (see Definition 11.6) with a success probability $\Omega(\frac{\varepsilon}{q_s})$ where q_s is the number of signature queries.

As in the previous case this adversary is also not allowed to choose freely which user he wants to corrupt. Another additional restriction here is that he has to come up with a level 0 signature. We can reduce this adversary $\mathcal{A}_3$ to our generic adversary $\mathcal{A}$ by requiring $\mathcal{A}$ to come up with a level 0 signature σ^* on behalf of a specific user i^* who is fixed in the beginning.

13.5.2. External Security. This notion captures that an attacker who is outside of the system is not able to corrupt users inside the system. In this definition the adversary is allowed to have access to all public keys, a signature oracle returning level 0 signatures on behalf *any* user and a re-signature oracle which returns re-signatures as defined above. In the end the adversary is required to come up with a level ℓ signature σ^* valid for a user $i^* \in \{0, \ldots, N-1\}$ of his choice. The adversary fails if he queried one of the oracles for a signature on behalf of user i^* on the forged message m^* before. The proof uses this adversary with advantage ε to solve a given ℓ-flexDH instance with a success probability $\Omega(\frac{\varepsilon}{N(q_s+q_{rs}}))$ where q_s is the number of signature queries and q_{rs} is the number of re-signature queries. Differing from the other proofs above the factor $\frac{1}{N}$ comes from the initial guess of the user i^* which will be corrupted by the adversary.

Compared to our generic adversary and the other adversaries in this security definition this adversary is the most limited one. We can reduce this adversary $\mathcal{A}_4$ to our generic adversary $\mathcal{A}$ by limiting $\mathcal{A}$'s access to the oracles $\mathcal{O}_{\text{Sign}}$ and $\mathcal{O}_{\text{ReSign}}$.

NOTE 13.2. *As the authors Libert & Vergnaud (2008a) point out correctly,*

the notion of external security "only makes sense if the re-signature keys are kept private by the proxy". *This means that the* private proxy property *is essential for the security definition of Ateniese & Hohenberger (2005). This was later also pointed out by Chow & Phan (2008) and Shao* et al. *(2010), see Remark 13.3 later.*

13.6. Results. The security definition from Ateniese & Hohenberger (2005) seems correct in practice because intuitively it covers all attack scenarios. However both Ateniese & Hohenberger (2005) and Libert & Vergnaud (2008a) lack of an argumentation of how this definition was constructed. Formally speaking this means that it is not clear if there are other attack scenarios or not. As already pointed out in the case of *delegatee security* there are some cases which are not considered in the original security definition. For example, in the case of *external security* adversaries who have access to more information such as some re-signature keys are also not considered.

On the other hand, using the query graph technique with a generic adversary allows us to avoid these artificial limitations of adversaries and splitting the security definition. Therefore it is safe to assume that our security definition is *not* equivalent to the original one. It rather encompasses the old one and also the unconsidered attack scenarios. Consequently we only need one proof instead of four.

As mentioned above, another point of critique is that the adversaries in the cases of *delegatee security* and *delegator security* are not allowed to choose the targeted user i^* freely. It seems that this determination of the targeted user i^* in these cases is somehow similar to the case of *external security*. Since in the proof *external security* the corrupted user i^* is guessed initially, the success probabilities of *delegatee security* and *delegator security* should at least differ from the success probability of *limited proxy security* because in that case the adversary is allowed to choose i^* freely.

13.7. Observations. This new security definition includes all requirements from Part II as we are going to show in the following.

13.7.1. Uni-directionality. Suppose that we have a signature scheme where the re-signature key R_{ij} can be used to translate signatures σ_i valid for user i into signatures σ_j valid for user j and vice versa. A generic adversary with access to the oracles from above can do the following:

1. Query $\mathcal{O}_{\text{Sign}}(i, m^*)$ to obtain a valid signature σ_i on m^* on behalf of user i.

2. Query $\mathcal{O}_{\text{ReKey}}(i^*, i)$ to get the re-signature key R_{i^*i}.

3. Return $\sigma_{i^*} \leftarrow \text{ReSign}(m^*, \cdot, \sigma_j, R_{i^*i}, X_{i^*}, X_i)$ which is a forged signature on m^* on behalf of user i^*.

This is a perfectly valid forgery in the sense of our new security definition. This results from the directed edges of the graph G_Q constructed from Q-list. The first query will create a path from [0] to $[i, m^*]$ the second query will create a path from $[i^*, m^*]$ to $[i, m^*]$ but there will be no path from [0] to $[i^*, m^*]$. Using an undirected graph instead, seems to make this security definition also valid for *bi-directional* schemes. However, it is not clear what impacts the usage of an undirected graph has on the other requirements.

13.7.2. Private proxy. Suppose that we have a signature scheme where the re-signature key R_{ij} is easily recovered from the signatures $\sigma_i(m)$ and $\sigma_j(m)$. An adversary with access to the oracles from above can do the following:

1. Query $\mathcal{O}_{\text{Sign}}(i, m)$ to get a valid signature $\sigma_i(m)$ on a message m on behalf of user i.

2. Query $\mathcal{O}_{\text{ReSign}}(\sigma_i, i, i^*, m)$ to get a valid re-signature $\sigma_{i^*}(m)$ on m on behalf of user i^* re-signed from $\sigma_i(m)$.

3. Recover R_{ii^*} from $\sigma_i(m)$ and $\sigma_{i^*}(m)$.

4. Query $\mathcal{O}_{\text{Sign}}(i, m^*)$ and get a valid signature $\sigma_i(m^*)$ on m^* on behalf of user i.

5. Return $\sigma^*_{i^*}(m^*) \leftarrow \text{ReSign}(m^*, \cdot, \sigma_i(m^*), R_{ii^*}, X_i, X_{i^*})$ which is a forged signature on the message m^* on behalf of user i^*.

As above this process is also a perfectly valid forgery in the sense of the new security definition because there will be no entry $[\mathcal{O}_{\text{ReKey}}, (i, i^*)]$ in Q-list. Therefore the graph algorithm will not be able to create an edge from the node $[i, m^*]$ to $[i^*, m^*]$. This means that our new security definition and also the original one only make sense if the proxy keeps the re-signature keys R_{ij} private (Chow & Phan 2008). However, we can modify the algorithm Query-graph such that it adds edges $([i, m], [j, m])$ for all messages $m \in M$ for every re-signature query entry $[\mathcal{O}_{\text{ReSign}}, (i, j, m)] \in Q$-list (step 15 in algorithm Query-graph). This means that the algorithm would then treat every entry $[\mathcal{O}_{\text{ReSign}}, (i, j, m)]$ in Q-list additionally as a re-signature key query, ie. $[\mathcal{O}_{\text{ReKey}}, (i, j)]$. This seems to make the security definition also valid for signature schemes with the *public proxy* property.

13.7.3. Transparency. Suppose that we have signature scheme where an adversary can distinguish between signed and re-signed signatures on the same level. This means that an algorithm isTransformed$(\cdot)$ is implicitly included in the system parameters which answers 1 if a signature was generated by the ReSign$(\cdot)$ algorithm (Chow & Phan 2008). Now we consider an adversary making the following queries:

1. Query $\mathcal{O}_{\text{Sign}}(i, m)$ to obtain a valid signature $\sigma_i(m)$ on a message m on behalf of user i.

2. Query $\mathcal{O}_{\text{ReSign}}(\sigma_i, i, j, m)$ to get a valid re-signature $\sigma_j(m)$ on m on behalf of user j re-signed from $\sigma_i(m)$.

If the adversary can somehow now extract some information from $\sigma_i(m)$ and $\sigma_j(m)$ that allows him to calculate the re-signature key R_{ij}, he can output valid forgery as shown above in the *private proxy* property. Besides that just the knowledge of isTransformed$(\sigma) = 1$, should be "safe" for the security of the signature scheme. Therefore, our new security definition also seems to apply for *non-transparent* proxy re-signature schemes.

13.7.4. Unlinkability. Similar to the *transparency* property this property is also highly related to the *private proxy* property. This means that the mere ability to link a re-signature to its predecessor does not seem to compromise the security of the signature scheme.

13.7.5. Non-transitivity. Suppose that we have transitive signature scheme where the re-signature key R_{ii^*} can easily be produced from R_{ij} and R_{ji^*}. Consider the following attack from Chow & Phan (2008):

1. Query $\mathcal{O}_{\text{Sign}}(i, m^*)$ to get σ_i on a message m on behalf of user i.

2. Query $\mathcal{O}_{\text{ReKey}}(i, j)$ to get R_{ij} the re-signature key from user i to j where $i \neq j$.

3. Query $\mathcal{O}_{\text{ReKey}}(j, i^*)$ to get R_{ij} the re-signature key from user i to j where $i \neq j \neq i^*$.

4. Compute R_{ii^*} from R_{ij} and R_{ii^*}.

5. Return $\sigma^* \leftarrow \text{ReSign}(m^*, \cdot, \sigma_i(m^*), R_{ii^*}, X_i, X_{i^*})$.

This is a valid forgery for the security definition of Ateniese & Hohenberger (2005). However, this is not the case in our new security definition. We consider

the query graph G_Q for this attack. The signature query $\mathcal{O}_{\text{Sign}}(i, m^*)$ will create a path from [0] to $[i, m^*]$ and the re-signature key queries for R_{ij} and R_{ji^*} will create paths from $[i, m^*]$ to $[j, m^*]$ and from $[j, m^*]$ to $[i^*, m^*]$ respectively. Since there is a path from [0] to $[i^*, m^*]$ in G_Q, this is *not* a valid forgery in the sense of our new security definition. Therefore our new security should also apply for *transitive* proxy re-signature schemes.

13.8. Multi-use and single-use. It seems that this new security definition is valid for both *single-use* and *multi-use* schemes since the algorithm Query-graph makes no distinction between signed and re-signed signatures. The nodes created by the algorithm only consider the ability of the adversary to generate *any* signature on a message m on behalf of user i and if he is able to translate *any* signature into his final output σ^* valid for user i^* on a message m^*. This means that in a *single-use* scheme a trivial forgery will be detected by the algorithm Query-graph but the new security definition may limit the adversary's capabilities.

13.9. Conclusions. The results and the observations from above lead us to the conclusion that our new security definition seems also to apply to proxy re-signatures with different requirements. The generic adversary and the query graph technique allow the necessary flexibility to prove the security of proxy re-signatures with *transitivity*, *transparency* and *linkability* properties. Slight modifications to the algorithm Query-graph seem to make our new security definition also useful for proxy re-signatures with *public proxy* and *bi-directionality* properties.

REMARK 13.3. *In the recent publication of Shao* et al. *(2010), the authors point out a "flaw" of the security definition of Ateniese & Hohenberger (2005) for uni-directional proxy re-signatures. The authors construct a uni-directional proxy re-signature scheme without the* private proxy *property which can be proven secure in the security definition of Ateniese & Hohenberger (2005) but suffers from an attack similar to the one above. The authors conclude that private proxy property is essential for the security definition of Ateniese & Hohenberger (2005), however as noted above this was already pointed out in Libert & Vergnaud (2008a) and also mentioned in Chow & Phan (2008) later. The authors Shao* et al. *(2010) also mention that the deficiency of the security model of Ateniese & Hohenberger (2005) is that this model* "tried to model *all* types of attacks on *all* types of proxy re-signatures" *and therefore is more complex than the security definitions for other types of signatures. The authors provide another security definition for uni-directional proxy re-signatures*

with *the private proxy property. However, the achieved results in this thesis let us believe that our new security definition for proxy re-signature schemes overcomes the mentioned complexity and is more simpler than both proposals. We also believe that our new security definition provides the necessary flexibility to be adapted for different types of proxy re-signature schemes as we have discussed in this section.*

14. Proof of security in the random oracle model

Using the new security definition we now prove that the signature scheme is secure in the random oracle model.

THEOREM 14.1. *The proxy re-signature scheme with L levels and N users is secure under the ℓ-flexDH assumption. More precisely, given an attacker $\mathcal{A}$ to Game 13.1 with advantage ε we can construct an algorithm $\mathcal{B}$ that solves an ℓ-flexDH instance for given $P, aP, bP \in \mathbf{G}$ with probability*

$$\Omega\left(q_{sk} \cdot q_{rk} \cdot \frac{(N - |I^*|)^3}{N^{|I^*|+3}(q_s + q_{rs})} \cdot \varepsilon\right).$$

where $a, b \in \mathbb{Z}_p^\times$, q_{sk} is the number of private key queries, q_{rk} is the number of re-signature key queries, q_s is the number of signature queries and q_{rs} the number of re-signature queries made by $\mathcal{A}$. Here $|I^*|$ denotes the cardinality of the set $I^* \subset \{0, \ldots, N-1\}$ which is chosen in advance by $\mathcal{B}$ as described below in the proof sketch.

Proof sketch: We construct an algorithm $\mathcal{B}$ which takes control of $\mathcal{A}$'s oracles, $\mathcal{O}_{\text{Skey}}$, $\mathcal{O}_{\text{ReKey}}$, $\mathcal{O}_{\text{Sign}}$ and $\mathcal{O}_{\text{ReSign}}$. We allow $\mathcal{A}$ to have access to a hashing oracle $\mathcal{O}_{\text{Hash}}$ which is also controlled by $\mathcal{B}$. All queries to $\mathcal{O}_{\text{Hash}}$ are stored in the Q_{Hash}-list while all other queries are stored in the Q-list which allows us to create the query graph G_Q. Figure 14.1 shows algorithm $\mathcal{B}$ using $\mathcal{A}$.

Note that in our new security definition $\mathcal{A}$ is allowed to forge a signature σ' on m^* on behalf of user $i' \in \{0, \ldots, N-1\}$ and then transform σ' *many* times with the corresponding re-signature keys into a signature σ^* on behalf of user $i^* \in \{0, \ldots, N-1\}$ before outputting it. If this is a valid forgery then there is no path from $[0]$ to $[i', m^*]$, but a path π from $[i', m^*]$ to $[i^*, m^*]$. Therefore, when $\mathcal{B}$ is challenged with $(P, aP, bP) \in \mathbf{G}$, he guesses in advance a set of users $I^* \subset \{0, \ldots, N-1\}$ which contains all users on the path π from $[i', m^*]$ to $[i^*, m^*]$. $\mathcal{B}$ sets the public keys X_i of users $i \in I^*$ as $z_i aP$, for some $z_i \xleftarrow{R} \mathbb{Z}_p^\times$

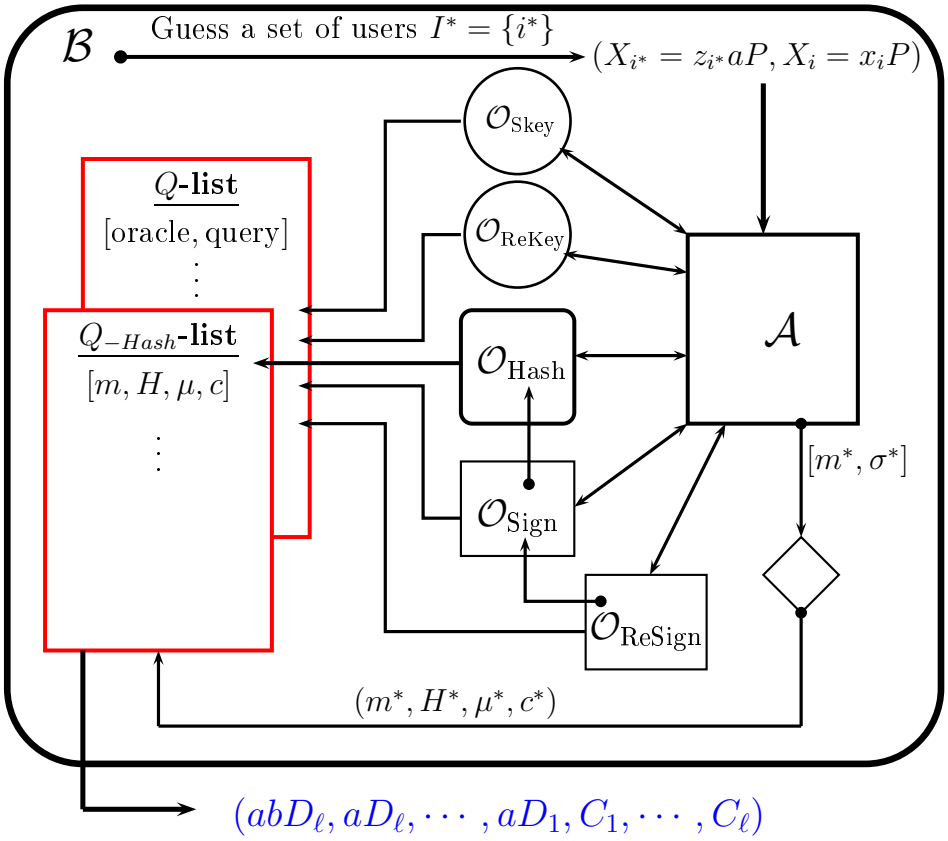


Figure 14.1: Algorithm $\mathcal{B}$ using $\mathcal{A}$

and all other public keys are set as $X_i = x_iP$ for some $x_i \xleftarrow{\$} \mathbb{Z}_p^\times$ and made available to $\mathcal{A}$. In the end when $\mathcal{A}$ comes up with a valid message signature pair (m^*, σ^*) valid for the public key X_{i^*}, $\mathcal{B}$ constructs the query graph G_Q from the Q-list and if this is a non-trivial forgery there will be no path from $[0]$ to $[i^*, m^*]$ in G_Q.

Proof. After guessing the set of users I^* and setting the public keys as described above $\mathcal{B}$ answers the oracle calls of $\mathcal{A}$ as follows:

Private key oracle $\mathcal{O}_{\text{Skey}}$ queries: When $\mathcal{A}$ asks $\mathcal{O}_{\text{Skey}}$ for the secret key of user i, $\mathcal{B}$ does the following:

Algorithm $\mathcal{O}_{\text{Skey}}$.
Input: A user $i \in \{0, \ldots, N-1\}$.

Output: The secret key x_i of user i or $\mathcal{B}$ aborts.

1. If $i \in I^*$ then $\mathcal{B}$ aborts.
2. Add $[\mathcal{O}_{\text{Skey}}, i]$ to Q-list.
3. Return x_i.

Re-signature key oracle $\mathcal{O}_{\text{ReKey}}$ queries: When $\mathcal{A}$ queries $\mathcal{O}_{\text{ReKey}}$ for re-signature keys, $\mathcal{B}$ does the following:

ALGORITHM $\mathcal{O}_{\text{ReKey}}$.

Input: Two users $i, j \in \{0, \ldots, N-1\}$.
Output: The re-signature key R_{ij} or $\mathcal{B}$ aborts.

1. If $i \notin I^*$ then
2. If $j \in I^*$ then $\mathcal{B}$ aborts.
3. Else $R_{ij} \leftarrow \frac{x_i}{x_j} P$.
4. Else
5. If $j \in I^*$ then $R_{ij} \leftarrow \frac{z_i a}{z_j a} P = \frac{z_i}{z_j} P$.
6. Else $R_{ij} \leftarrow \frac{z_i a}{x_j} P$.
7. Add $[\mathcal{O}_{\text{ReKey}}, (i, j)]$ to Q-list.
8. Return R_{ij}.

Hashing oracle $\mathcal{O}_{\text{Hash}}$ queries: When $\mathcal{A}$ asks for the hash value of a message m, $\mathcal{B}$ runs the following algorithm using the global hash list Q_{Hash}-list. This list consists of 4-tuple entries $[m, H, \mu, c]$, where m is the message, H the answer to the query, $\mu \xleftarrow{R} \mathbb{Z}_p^{\times}$ a randomly chosen parameter and $c \xleftarrow{R} \{0, 1\}$ a random bit with probability $Pr[c = 0] = \zeta$ for a $\zeta \in (0; 1)$ to be adopted later.

ALGORITHM $\mathcal{O}_{\text{Hash}}$.

Input: A message $m \in \{0, 1\}^*$.
Output: A hash value H.

1. If $m \in Q_{\text{Hash}}$-list then
2. $[m, H, \mu, c] \leftarrow Q_{\text{Hash}}$-list.
3. Else
4. Generate a bit $c \xleftarrow{R} \{0, 1\}$.
5. Choose a random $\mu \xleftarrow{R} \mathbb{Z}_p^{\times}$.
6. If $c = 1$ then

7. $H \leftarrow \mu bP$.
8. Else $H \leftarrow \mu P$.
9. Add $[m, H, \mu, c]$ to Q_{Hash}-list.
10. Return H.

Signature oracle $\mathcal{O}_{\text{Sign}}$ queries: When $\mathcal{A}$ queries $\mathcal{O}_{\text{Sign}}$ for a signature of user i on a message m, $\mathcal{B}$ runs the following algorithm

ALGORITHM $\mathcal{O}_{\text{Sign}}$.

Input: A message $m \in \{0,1\}^*$, a user $i \in \{0, \ldots, N-1\}$, a desired level $\ell \in \{0, \ldots, L-1\}$.

Output: A level ℓ signature $\sigma^{(\ell)}$ on m valid for the public key X_i or $\mathcal{B}$ aborts.

1. Run algorithm $\mathcal{O}_{\text{Hash}}(m)$.
2. $[m, H, \mu, c] \leftarrow Q_{\text{Hash}}$-list.
3. If $c = 1$ then $\mathcal{B}$ aborts.
4. If $i \in I^*$ then $\sigma^{(0)} \leftarrow \mu z_i a P$.
5. Else $\sigma^{(0)} \leftarrow x_i H$.
6. If $\ell > 0$ then
7. For $k = 1, \ldots, \ell$ do
8. $\sigma^{(k)} \leftarrow \boxed{\texttt{ADD TRIVIAL H}} \blacktriangleleft \sigma^{(k-1)}$.
9. $\sigma^{(\ell)} \leftarrow \boxed{\texttt{RE-RANDOM}} \blacktriangleleft \sigma^{(\ell)}$.
10. Add $[\mathcal{O}_{\text{Sign}}, (i, m)]$ to Q-list.
11. Return $\sigma^{(\ell)}$.

Re-signature oracle $\mathcal{O}_{\text{ReSign}}$ queries: When $\mathcal{A}$ queries $\mathcal{O}_{\text{ReSign}}$ for the re-signature of $\sigma^{(\ell-1)}$ valid for X_i from user i to j, $\mathcal{B}$ ignores this and uses $\mathcal{O}_{\text{Sign}}$ to create $\sigma^{(\ell)}$ a level ℓ signature for user j. The resulting signature is then returned to $\mathcal{A}$. Namely:

ALGORITHM $\mathcal{O}_{\text{ReSign}}$.

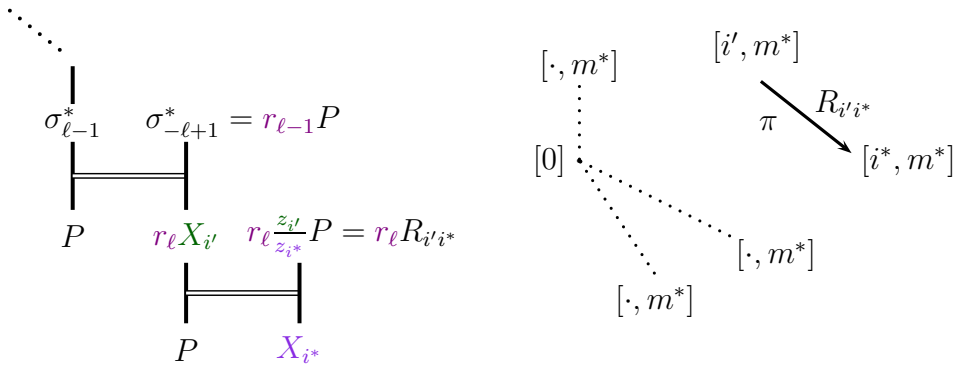
Input: Two users keys (i, j), a message m and a level $\ell - 1$ signature $\sigma^{(\ell-1)}$ on m valid for X_i.

Output: A level ℓ signature $\sigma^{(\ell)}$ on m valid for X_j or $\mathcal{B}$ aborts.

1. $\sigma^{(\ell)} \leftarrow \mathcal{O}_{\text{Sign}}(m, j, \ell)$.
2. Add $[\mathcal{O}_{\text{ReSign}}, (i, j, m)]$ to Q-list.
3. Return $\sigma^{(\ell)}$.

Note that the call of algorithm $\mathcal{O}_{\text{Sign}}$ in step 1 can cause an abort which is not handled here since $\mathcal{B}$ is in control and knows when to abort.

Final output: Finally $\mathcal{A}$ outputs a message signature pair (m^*, σ^*) where $\sigma^* = (\sigma_0^*, \ldots, \sigma_\ell^*, \sigma_{-\ell}^*, \ldots, \sigma_{-1}^*)$ is a valid level ℓ signature on m^* on behalf of user $i^* \in I^*$. If initially $\mathcal{B}$ guessed the set I^* correctly and did not have to abort before, he runs the algorithm FINALIZE below. Here $\mathcal{B}$ creates the query graph G_Q from the Q-list with this algorithm and finds the path π in G_Q starting at $[i', m^*]$ and ending at $[i^*, m^*]$ to determine the initially forged user $i' \in I^*$ and all the re-signature keys leading to the final forgery σ^*. To understand what happens at the final step consider the following example shown in Figure 14.2.



(a) Forged and transformed signature σ^* (b) The query graph G_Q

Figure 14.2: A forged signature with the corresponding query graph

In this example $\mathcal{A}$ forged a signature on behalf of user $i' \in I^*$ and transformed it only once into σ^* on behalf of user $i^* \in I^*$ with the re-signature key $R_{i'i^*}$. Figure 14.2(a) shows the the appended H where σ_ℓ^* is $r_\ell z_{i'} aP$ and $\sigma_{-\ell}^*$ is $r_\ell R_{i'i^*}$. Now, $\mathcal{B}$ needs to remove $z_{i'}$ from σ_ℓ and $\frac{z_{i'}}{z_{i^*}}$ from $\sigma_{-\ell}$ to achieve a "regular" signature on behalf of user i^*. For this $\mathcal{B}$ finds the path π from $[i', m^*]$ to $[i^*, m^*]$ in the corresponding query graph G_Q shown in Figure 14.2(b). Here in this example $\mathcal{B}$ determines the length of the path $|\pi|$ as 1 and knows that the output signature was only transformed once. Therefore $\mathcal{B}$ determines the order of the users $\pi_0 = i'$ and $\pi_1 = i^*$ and calculates the elements $\frac{1}{z_{\pi_0}} = \frac{1}{z_{i'}}$ and $\frac{1}{z_{\pi_1}} = \frac{1}{z_i^*}$. This allows him to remove the unwanted elements and treat σ^* as if it was never transformed. Now consider the following algorithm:

FINALIZE.

Input: A message m^*, a signature σ^* valid for the public key X_{i^*} on m^*, the hash list Q_{Hash}-list and the general query list Q-list.

Output: An ℓ-flexDH instance $(abD_\ell, aD_\ell, \ldots, aD_1, C_\ell, \ldots, C_1)$ or $\mathcal{B}$ aborts.

1. Create the query graph $G_Q \leftarrow$ Query-graph$(i^*, m^*, Q\text{-list})$.
2. If there is a path from $[0]$ to $[i^*, m^*]$ in G_Q then $\mathcal{B}$ aborts.
3. Run algorithm $\mathcal{O}_{\text{Hash}}(m^*)$.
4. $[m^*, H^*, \mu^*, c^*] \leftarrow Q_{\text{Hash}}$-list.
5. If $c^* = 0$ then $\mathcal{B}$ aborts.
6. Find the path π from $[i', m^*]$ to $[i^*, m^*]$ with length $|\pi| = k$.
7. Determine the order of users $\pi_0 = i', \ldots, \pi_k = i^*$ on the path π.
8. Calculate the elements $\frac{1}{z_{\pi_0}} = \frac{1}{z_{i'}}, \frac{1}{z_{\pi_1}}, \ldots, \frac{1}{z_{\pi_k}} = \frac{1}{z_{i^*}}$.
9. Return $\Bigg(\left(\frac{1}{z_{\pi_0}\mu^*}\right)\sigma_0^*, (\frac{1}{z_{\pi_0}})\sigma_1^*, \ldots, (\frac{1}{z_{\pi_0}})\sigma_{\ell-k}^*, \left(\frac{1}{z_{\pi_1}}\right)\sigma_{\ell-k+1}^*,$
$\left(\frac{1}{z_{\pi+2}}\right)\sigma_{\ell-k+2}^*, \ldots, \left(\frac{1}{z_{\pi_k}}\right)\sigma_\ell^*,$
$\left(\frac{z_{\pi_k}}{z_{\pi_{k-1}}}\right)\sigma_{-\ell}^*, \left(\frac{z_{\pi_{k-1}}}{z_{\pi_{k-2}}}\right)\sigma_{-\ell+1}^*, \left(\frac{z_{\pi_{k-2}}}{z_{\pi_{k-3}}}\right)\sigma_{-\ell+2}^*, \ldots, \left(\frac{z_{\pi_1}}{z_{\pi_0}}\right)\sigma_{-\ell+k}^*,$
$\sigma_{-\ell+k+1}^*, \ldots, \sigma_{-2}^*, \sigma_{-1}^* \Bigg)$.

Recall that $\mathcal{B}$ initially guessed a set of users $I^* \subset \{0, \ldots, N\}$ whose public keys he set as $X_i = z_i aP$ for some $z_i \xleftarrow{\$} \mathbb{Z}_p^\times$. In the example from Figure 14.2, the set I^* contains the two users $\{i', i^*\}$ and thus, the algorithm FINALIZE would return

$$\left(\left(\frac{1}{z_{\pi_0}\mu^*}\right)\sigma_0^*, (\frac{1}{z_{\pi_0}})\sigma_1^*, \ldots, (\frac{1}{z_{\pi_0}})\sigma_{\ell-1}^*, \left(\frac{1}{z_{\pi_1}}\right)\sigma_\ell^*, \left(\frac{z_{\pi_1}}{z_{\pi_0}}\right)\sigma_{-\ell}^*, \sigma_{-\ell+1}^*, \ldots, \sigma_{-2}^*, \sigma_{-1}^*\right)$$

in step 9. Where this value can be written as

$$(abD_\ell, aD_\ell, \ldots, aD_1, C_\ell, \ldots, C_1),$$

which is a valid ℓ-flexDH instance because for all $j \in \{1, \ldots, \ell\}$ we have $\log_P D_j = \prod_{i=1}^{j} \log_P C_i$ and further C_j is not the neutral element of the group $\mathbf{G}$.

In the simplest case where $I^* = \{i^*\}$, $\mathcal{A}$ outputs a forged signature which was not transformed. In this case the return value is

$$\left(\left(\frac{1}{z_{i^*}\mu^*}\right)\sigma_0^*, (\frac{1}{z_{i^*}})\sigma_1^*, \ldots, (\frac{1}{z_{i^*}})\sigma_\ell^*, \sigma_{-\ell}^*, \sigma_{-\ell+1}^*, \ldots, \sigma_{-1}^*\right) =$$

$$= \left(\left(\frac{1}{z_{i^*}\mu^*} \right) (r_\ell \cdots r_1) z_{i^*} \mu^* abP, (\frac{1}{z_{i^*}})(r_\ell \cdots r_1) z_{i^*} aP, \ldots, (\frac{1}{z_{i^*}}) r_\ell z_{i^*} aP, \sigma^*_{-\ell}, \ldots, \sigma^*_{-1} \right).$$

Here we can easily see that this is a valid ℓ-flexDH instance as

$$(abD_\ell, aD_\ell, \ldots, aD_1, C_\ell, \ldots, C_1),$$

with $D_i = r_\ell \cdots r_i P$ and $C_i = r_i P$ and for all $j \in \{1, \ldots, \ell\}$ we have $\log_P D_j = \prod_{i=1}^{j} \log_P C_i$. Thus, the return value in step 9 of algorithm FINALIZE is a valid ℓ-flexDH instance as

$$(abD_\ell, aD_\ell, \ldots, aD_1, C_\ell, \ldots, C_1),$$

where for all $j \in \{1, \ldots, \ell\}$ we have $\log_P D_j = \prod_{i=1}^{j} \log_P C_i$ and further C_j is not the neutral element of the group $\mathbf{G}$.

Note that it is possible that different paths π exist which may even have common edges. In this case $\mathcal{B}$ has to try out all paths π until he finds a valid ℓ-flexDH instance. For simplicity reasons we assume that there is only one such path π.

The success probability of $\mathcal{B}$. We use an analysis similar to that in Coron (2000) to determine a lower bound for the success probability of $\mathcal{B}$. Remember that initially $\mathcal{B}$ takes a guess of the set of users $I^* \subset \{0, \ldots, N-1\}$. The probability that $\mathcal{B}$ guesses the correct set I^* is

$$\frac{1}{N} \cdot \frac{1}{N-1} \cdots \frac{1}{N-|I^*|+1} = \frac{1}{\binom{N}{|I^*|}}.$$

$\mathcal{A}$ asks for the private key of user $i \in I^*$ with probability $\frac{|I^*|}{N}$, thus the probability that $\mathcal{B}$ does *not* abort for q_{sk} many private key queries of $\mathcal{A}$ is

$$q_{sk} \frac{N - |I^*|}{N}.$$

$\mathcal{B}$ aborts the simulation for any re-signature key query R_{ij} of $\mathcal{A}$ if $i \notin I^*$ and $j \in I^*$. This happens for a single re-signature key query of $\mathcal{A}$ with probability

$$\frac{N - |I^*|}{N} \cdot \frac{|I^*|}{N-1}.$$

Thus, the probability that $\mathcal{B}$ does not abort for q_{rk} many re-signature key queries of $\mathcal{A}$ is

$$q_{rk} \frac{1}{R} := q_{rk} \left(1 - \frac{N - |I^*|}{N} \cdot \frac{|I^*|}{N-1} \right).$$

Recall that a query to $\mathcal{O}_{\text{ReSign}}$ triggers actually a signature query to $\mathcal{O}_{\text{Sign}}$. Recall also that $\mathcal{O}_{\text{Sign}}$ uses $\mathcal{O}_{\text{Hash}}$ to answer the signature queries. When $\mathcal{O}_{\text{Hash}}$ generates a 4-tuple $[m, H, \mu, c]$, $\mathcal{O}_{\text{Sign}}$ causes $\mathcal{B}$ to abort if $c = 1$. Therefore the probability that $\mathcal{B}$ answers to *all* signature and re-signature queries of $\mathcal{A}$ and does not abort here is $\zeta^{q_s+q_{rs}}$ since the probability $Pr[c = 0] = \zeta$. In the end when $\mathcal{A}$ outputs a valid message signature pair (m^*, σ^*) the bit $c^* = 1$ happens with probability $Pr[c^* = 1] = 1 - \zeta$. This means that, if $\mathcal{B}$ guessed the set of users I^* correctly and did not have to abort because of a secret key or a re-signature key query of $\mathcal{A}$, the probability that he outputs an ℓ-flexDH answer is at least

$$\alpha(\zeta) = \zeta^{q_s+q_{rs}} \cdot (1 - \zeta).$$

The function $\alpha(\zeta)$ is maximal for $\zeta_{max} = 1 - \frac{1}{q_s+q_{rs}+1}$ which gives us

$$\alpha(\zeta_{max}) = \frac{1}{q_s + q_{rs}} \cdot \left(1 - \frac{1}{q_s + q_{rs} + 1}\right)^{q_s+q_{rs}+1}.$$

Then the success probability of $\mathcal{B}$ is

$$Pr[\mathcal{B} \text{ is successful }] \geq \varepsilon \cdot q_{sk} \frac{N - |I^*|}{N} \cdot \frac{1}{\binom{N}{|I^*|}} \cdot \frac{q_{rk}}{R} \cdot \frac{1}{q_s + q_{rs}} \left(1 - \frac{1}{q_s + q_{rs} + 1}\right)^{(q_s+q_{rs}+1)}.$$

Now we notice that

$$\frac{1}{\binom{N}{|I^*|}} \geq \frac{1}{N^{|I^*|}}$$

and also

$$q_{rk} \frac{1}{R} \geq q_{rk} \cdot \frac{(N - |I^*|)^2}{N^2}.$$

This gives us the announced bound of

$$\Omega\left(q_{sk} \cdot q_{rk} \cdot \frac{(N - |I^*|)^3}{N^{|I^*|+3}(q_s + q_{rs})} \cdot \varepsilon\right).$$

□

14.1. Results. We emphasize here that the partitioning $I^* \subset \{0, \ldots, N-1\}$ of the users is necessary because we allow $\mathcal{A}$ to translate his actual forgery before outputting it. This results from the fact that in our new security definition we grant $\mathcal{A}$ access to re-signature keys which do not necessarily lead to a trivial forgery. Consequently in this reduction the size of I^* effects the success probability of $\mathcal{B}$. Now we analyze how the size of I^*, the number of secret key queries

q_{sk} and the number of re-signature key queries q_{rk} effect the success probability of $\mathcal{B}$. Firstly, we notice that the factor

$$\frac{(N - |I^*|)^3}{N^{|I^*|+3}}$$

can grow very fast and is the strongest decisive factor in this reduction. Therefore we require the size of I^* to be bounded and note this in the following corollary.

COROLLARY 14.2 (Size of $|I^*|$). *In order to have a non-negligible success probability of $\mathcal{B}$, $N^{|I^*|+3}$ is required to be polynomial in N. Therefore, we require $|I^*| \in \mathrm{O}(1)$.*

We note the following corollary for the smallest size of I^*.

COROLLARY 14.3 (Case: $|I^*| = 1$). *In the case where $I^* = \{i^*\}$ we have the initial probability of $\frac{1}{N}$. The probability that $\mathcal{B}$ does not abort for a single secret key query $\frac{N-1}{N}$. The probability of $\mathcal{B}$ not aborting for a single re-signature key query is also $\frac{N-1}{N}$. This gives us the success probability of $\mathcal{B}$ as*

$$\Omega\left(q_{sk} \cdot q_{rk} \cdot \frac{(N-1)^2}{N^3(q_s + q_{rs})} \cdot \varepsilon\right).$$

In this case we can omit the factor related to the secret key queries of $\mathcal{A}$ because even if $\mathcal{B}$ could answer query for the secret key of user i^* this would create a path from $[0]$ and $[i^*, m^*]$ in G_Q and cause $\mathcal{B}$ to abort anyway.

Corollary 14.3 leads us to the conclusion that if $\mathcal{B}$ initially choose the "correct" set I^* such that all the users of I^* are on the path π, the abortion for secret key queries is "justified" and the factor related to q_{sk} many secret key queries of $\mathcal{A}$ can be omitted, since these would lead to a trivial forgery anyway. We note this in the following corollary.

COROLLARY 14.4 ($|I^*| = |\pi| + 1$). *We require that $\mathcal{B}$ initially chooses the correct set I^* such that all users in I^* are on the path π, ie. $|I^*| = |\pi| + 1$. Then we can omit the factor related to q_{sk} many secret key queries of $\mathcal{A}$ since the queries for secret keys of users $i \in I^*$ would necessarily lead to a trivial forgery and cause the simulation to abort anyway. This gives us the success probability of $\mathcal{B}$ as*

$$\Omega\left(q_{rk} \cdot \frac{(N - |I^*|)^2}{N^{|I^*|+2}(q_s + q_{rs})} \cdot \varepsilon\right).$$

We also notice that the factor related to the re-signature key queries of $\mathcal{A}$ is another decisive factor for the success probability of $\mathcal{B}$. In this context we note the following corollary.

COROLLARY 14.5 (Case $\mathcal{O}_{\text{ReKey}}$ abortions forbidden). *If the adversary is forbidden to ask for re-signature keys R_{ij} of users $i \notin I^*$ and $j \in I^*$ the success probability of $\mathcal{B}$ is completely independent of the factor*

$$q_{rk}\frac{1}{R} := q_{rk}\left(1 - \frac{N - |I^*|}{N} \cdot \frac{|I^*|}{N-1}\right).$$

Therefore the success probability of $\mathcal{B}$ would be

$$\Omega\left(\frac{\varepsilon}{N^{|I^*|}(q_s + q_{rs})}\right).$$

Recalling Section 13.5, we now want use the adversaries $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ from the original security definition in our reduction and compare the results.

COROLLARY 14.6 (Limited proxy security). *The adversary $\mathcal{A}_1$ in case of limited proxy security has access to all re-signature keys and a signing oracle and returns a valid level ℓ signature on behalf of a user $i^* \in \{0, \ldots, N-1\}$. $\mathcal{B}$ can provide $\mathcal{A}_1$ with this information by setting $|I^*| = N$. This means that all public keys X_i are set as $z_i aP$ for some $z_i \xleftarrow{\$} \mathbb{Z}_p^\times$ for all users $i \in \{0, \ldots, N-1\}$. This allows $\mathcal{B}$ to answer* all *re-signature key queries R_{ij} of $\mathcal{A}_1$ without aborting the simulation. $\mathcal{B}$ handles the signature queries of $\mathcal{A}_1$ with his signature algorithm $\mathcal{O}_{\text{Sign}}$. In the end when $\mathcal{A}_1$ comes up with a level ℓ forgery σ^*, $\mathcal{B}$ treats σ^* as if it was never translated and retrieves the ℓ-flexDH instance from σ^*. The success probability of $\mathcal{B}$ is the same as it is in the original security definition. Namely,*

$$\Omega\left(\frac{\varepsilon}{q_s}\right).$$

Note here that $\mathcal{A}_1$ does not make any secret key and re-signature queries such that the factors q_{sk} and q_{rs} are omitted.

COROLLARY 14.7 (Delegatee security). *The adversary $\mathcal{A}_2$ in case of delegatee security targets a specified delegatee i^* and has access to a signing oracle which provides him with signatures on behalf of i^*. $\mathcal{A}_2$ has also access to all secret keys x_i for users $i \in \{0, \ldots, N-1\} \setminus \{i^*\}$ and to all re-signatures keys R_{ij} except R_{ii^*} for any $i \neq i^*$. $\mathcal{B}$ can provide $\mathcal{A}_2$ with this information by choosing*

$I^* = \{i^*\}$. Since $\mathcal{A}_2$ targets a specified user i^* this can be done with any *user $i \in \{0, \ldots, N-1\}$, therefore this is not an initial guess as it is in our security definition. This means also that all the secret key and all the re-signature key queries made by $\mathcal{A}_2$ can be answered because by assumption $\mathcal{A}_2$ is not allowed to ask for the secret key of user i^* or any re-signature key R_{ii^*}. The signature queries of $\mathcal{A}_2$ are handled by $\mathcal{B}$ as usual. In the end when $\mathcal{A}_1$ comes up with a level ℓ forgery σ^*, $\mathcal{B}$ retrieves an ℓ-flexDH instance with a success probability of*

$$\Omega\left(\frac{\varepsilon}{q_s}\right).$$

This is the same as it is in the original security definition in the case of delegatee security. Again, the factor q_{rs} is omitted since $\mathcal{A}_2$ does not ask for re-signatures.

COROLLARY 14.8 (Delegator security). *The adversary $\mathcal{A}_3$ in the case of delegator security targets a specified delegator i^* and has access to a signing oracle which provides him with signatures on behalf of i^*. $\mathcal{A}_2$ has also access to all secret keys x_i for users $i \in \{0, \ldots, N-1\}$ including R_{i^*i} and R_{ii^*}. In the end $\mathcal{A}_3$ comes up with a level 0 signature on behalf of user i^*. $\mathcal{B}$ can answer the signature and secret key queries of $\mathcal{A}_3$ after specifying $I^* = \{i^*\}$ for a user $i^* \in \{0, \ldots, N-1\}$ but he cannot answer the re-signature key queries R_{ii^*} of $\mathcal{A}_3$ in this setting. In the end when $\mathcal{A}_3$ comes up with a level 0 forgery with a slight modification to $\mathcal{B}$'s algorithm* FINALIZE, *the success probability of $\mathcal{B}$ would then be*

$$\Omega\left(q_{rk} \cdot \frac{\varepsilon}{Nq_s}\right),$$

for q_{rk} many re-signature key queries of $\mathcal{A}_3$. However, we note that the reduction in the original security definition is done under the mCDH assumption (Definition 11.6) where an additional element $\frac{1}{a}P$ is available to $\mathcal{B}$. A slight modification to the re-signature key algorithm $\mathcal{O}_{\text{ReKey}}$ of $\mathcal{B}$ with the element $\frac{1}{a}P$ would allow him to answer all re-signature key queries of $\mathcal{A}_3$ including R_{ii^*}. This then gives us the success probability of

$$\Omega\left(\frac{\varepsilon}{q_s}\right),$$

which is the same as it is in the original security definition.

COROLLARY 14.9 (External security). *The adversary $\mathcal{A}_4$ in the case of external security has access to a signature and a re-signature oracle, and returns a valid level ℓ signature on behalf of a user $i^* \in \{0, \ldots, N-1\}$. $\mathcal{B}$ can provide $\mathcal{A}_4$ with this information by initially guessing a user i^* and setting $I^* = \{i^*\}$. $\mathcal{B}$ then can answer all signature and re-signature key queries of $\mathcal{A}_4$ with algorithm $\mathcal{O}_{\text{Sign}}$ and algorithm $\mathcal{O}_{\text{ReSign}}$, respectively. In the end when $\mathcal{A}_4$ outputs a level ℓ forgery on behalf of user i^*, $\mathcal{B}$ retrieves an ℓ-flexDH instance with a success probability of*

$$\Omega\left(\frac{\varepsilon}{N(q_s + q_{rs})}\right).$$

This is the same as it is in the original security definition in the case of external security. Note that $\mathcal{A}_4$ does neither ask for secret keys nor for re-signature keys.

As we can see there is a "huge" gap between the success probabilities of our reduction and the reductions done with the adversaries to the original security definition. A strategy to overcome this gap would be to force our generic adversary to output a non-transformed signature σ^* which would be the case discussed in Corollary 14.3. This can be done for example by accessing the adversaries memory and retrieving the actual forgery σ' but this requires that the adversary keeps σ' in his memory until it outputs σ^*. Similarly we could stop the adversary at the time of the actual forgery by monitoring its memory and retrieve σ'. However, both cases require access to the adversary's memory. Ideally, if we could force the adversary to output a non-transformed signature and forbid its access to re-signature keys which cause $\mathcal{B}$ to abort, we could obtain a combination of the cases discussed in Corollary 14.3 and Corollary 14.5. This would give us the same success probability as in Corollary 14.9 but it seems that we cannot limit the adversary such that it does not ask for "bad" re-signature keys.

Another strategy to reduce this gap would be to to set all public keys X_i as $z_i aP$ for some $z_i \xleftarrow{\$} \mathbb{Z}_p^\times$ for all users $i \in \{0, \ldots, N-1\}$ as we did in Corollary 14.6. $\mathcal{B}$ then can answer all re-signature key queries but he is not able to provide the adversary $\mathcal{A}$ with any secret key information. Therefore $\mathcal{B}$ would have to abort for *any* secret key query of $\mathcal{A}$ which can be avoided if $\mathcal{A}$'s access to secret keys is forbidden. In this case the success probability of $\mathcal{B}$ would be $\Omega\left(\frac{\varepsilon}{(q_s+q_{rs})}\right)$ which is similar to the results in the cases of Corollary 14.6, Corollary 14.7 and Corollary 14.8.

A different strategy to reduce this gap would be to change the underlying cryptographic assumption. As in Corollary 14.8, if the additional element $\frac{1}{a}P$ was also available to $\mathcal{B}$, he can answer *all* re-signature key queries of $\mathcal{A}$. In this

case the success probability of $\mathcal{B}$ would be

$$\Omega\left(\frac{\varepsilon}{N^{|I^*|}(q_s+q_{rs})}\right),$$

since we can omit the factor related to q_{sk} many secret key queries. Then the underlying cryptographic assumption would be a combination of the mCDH and the ℓ-flexDH problem. We call this the *modified ℓ-flexible Diffie-Hellman* (m-ℓ-flexDH) problem and note this in the following definition.

DEFINITION 14.10. *The* ***modified ℓ-flexible Diffie-Hellman*** *(m-ℓ-flexDH) problem is, given $P, aP, \frac{1}{a}P, bP \in \boldsymbol{G}$ to find a $(2\ell+1)$-tuple*

$$(abD_\ell, aD_\ell, \ldots, aD_1, C_\ell, \ldots, C_1) \in \boldsymbol{G}^{2\ell+1}$$

such that $\log_P D_j = \prod_{i=1}^{j} \log_P C_i$ for all $i \in \{1, \ldots, \ell\}$, where C_i is not the neutral element of the group $\mathbf{G}$.

Although it seems that the m-ℓ-flexDH problem is hard to solve, an adaptation of the proof of hardness for the ℓ-flexDH problem provided by Libert & Vergnaud (2008a) does not seem to work here. Thus, we cannot be certain that the reduction would be valid in this case.

We conclude from these results that the huge gap between the success probabilities results primarily from the generic adversary in our security definition. It seems that there is no strategy to overcome this gap between the success probabilities without artificially limiting our generic adversary's capabilities. On the other hand, these artificial limitations of the adversaries in the original security definition motivated us to construct a new security definition in which the adversary has access to as much information as possible. It seems that our new security definition is more "strict" compared to the old one and therefore we end up with a smaller success probability in the reduction. However, we firmly believe that this reduction in our new security definition provides a more concrete estimation of the success probability.

15. The signature scheme in the standard model

In this section we use a trick from Waters (2005) to eliminate the random oracle and instantiate the hash function H by a certain collision resistant hash function. A slight modification of the signature scheme will allow us to prove the security of the signature scheme also in the standard model. Note here that the common public parameters have to be generated by a trusted third party which remains off-line after the setup phase.

Setup(λ, n): On input of the security parameter λ and the length n of the messages to be signed, this algorithm chooses bilinear groups $(\mathbf{G}, \mathbf{G}_T)$ of prime order $p > 2^\lambda$, two generators $P, Q \xleftarrow{\$} \mathbf{G}$ and a random vector $\vec{u} = (U', U_1, \ldots, U_n) \xleftarrow{\$} \mathbf{G}^{n+1}$ vector of length $(n+1)$.

The vector $\vec{u}$ defines a function $H\colon \{0,1\}^n \to \mathbf{G}$ which maps n-bit strings $m = m_1, \ldots, m_n$ to $\mathbf{G}$ as $H(m) = U' + \sum_{i=1}^{n} m_i U_i$ where $m_i \in \{0,1\}$.

The public parameters are:

$$\mathbf{cp} = \{\lambda, n, \mathbf{G}, \mathbf{G}_T, P, H, \vec{u}\}.$$

Keygen(cp): This algorithm outputs user i's public and private key pair (X_i, x_i) for a random $x_i \xleftarrow{\$} \mathbb{Z}_p^\times$ and $X_i = x_i P$.

ReKeygen(cp, X_i, x_j): Given the public key X_i of user i and the private key x_j of user j this algorithm outputs the re-signature key as $R_{ij} = \frac{1}{x_j} X_i = \frac{x_i}{x_j} P$.

Verify(cp, $m, 0, \sigma, X_i$): The validity of a level 0 signature $\sigma = (\sigma_0, \sigma_\infty)$ on a message $m \in \{0,1\}^n$ for the public key X_i is verified if the following equation holds

$$e(\sigma_0, P) \stackrel{?}{=} e(X_i, Q) \cdot e(\sigma_\infty, H(m)). \tag{15.1}$$

The algorithm returns 1 if the input signature is valid and 0 otherwise.

Verify(cp, $m, \ell, \sigma^{(\ell)}, X_i$): The validity of a level ℓ signature

$$\sigma^{(\ell)}(m) = (\sigma_0, \sigma_1, \ldots, \sigma_\ell, \sigma_{-\ell}, \ldots, \sigma_{-1}, \sigma_\infty) \in \mathbf{G}^{2\ell+2}$$

on a message $m \in \{0,1\}^n$ for the public key X_i, is verified by the following $\ell + 1$ equations

$$e(\sigma_0, P) \stackrel{?}{=} e(\sigma_1, Q) \cdot e(\sigma_\infty, H(m)), \tag{15.2}$$

$$e(\sigma_k, P) \stackrel{?}{=} e(\sigma_{k+1}, \sigma_{-k}) \quad \text{for } k \in \{1, \ldots, \ell - 1\}, \tag{15.3}$$

$$e(\sigma_\ell, P) \stackrel{?}{=} e(X_i, \sigma_{-\ell}). \tag{15.4}$$

The algorithm returns 1 if the input signature is valid and 0 otherwise.

H-notation for the modified signature scheme. Before continuing to the signing and re-signing algorithms we want to make a graphical connection between the signature elements and picture them in the H-notation as we did in Part II. The verification equation (15.1) states that the elements of a modified level 0 signature valid for the public key X_i on a message m are connected as shown in Figure 15.1.

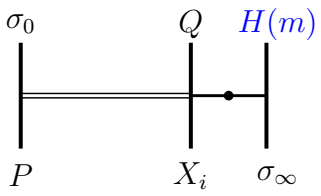


Figure 15.1: A modified level 0 signature

We deduce from Figure 15.1 the signing process at level 0 as follows.

Sign(cp, $m, 0, x_i$**):** On input of a message $m \in \{0,1\}^n$ and the private key x_i of signer i, this algorithm chooses a random $t \xleftarrow{R} \mathbb{Z}_p^\times$ and outputs a level 0 signature as

$$\sigma^{(0)}(m) = (\sigma_0, \sigma_\infty) = (x_i Q + tH(m), tP).$$

Using Figure 15.1 as a basis and interpreting the verification equations (15.2), (15.3) and (15.4) we connect the elements of a modified level ℓ signature valid for X_i on m as show in Figure 15.2.

Now we deduce from Figure 15.2 the signing process at level ℓ as follows.

Sign(cp, m, ℓ, x_i**):** On input a message $m \in \{0,1\}^n$ and the private key x_i of signer i and $\ell \in \{1, \ldots, L\}$ the level of the signature, this algorithm chooses $\ell + 1$ random coefficients $t, r_1, r_2, \ldots, r_\ell \xleftarrow{R} \mathbb{Z}_p^\times$ and outputs

$$\sigma^{(\ell)}(m) = (\sigma_0, \sigma_1, \ldots, \sigma_\ell, \sigma_{-\ell}, \ldots \sigma_{-2}, \sigma_{-1}, \sigma_\infty) \in \mathbf{G}^{2\ell+2}$$

with:

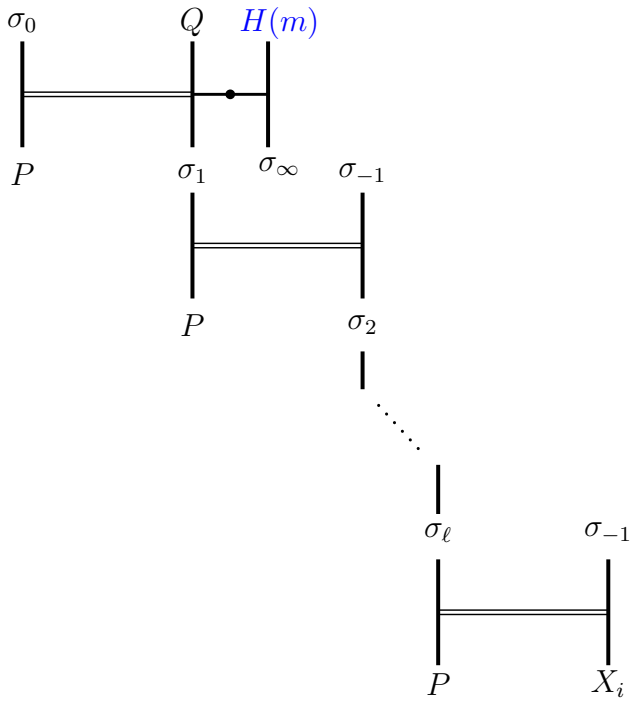


Figure 15.2: A modified level ℓ signature

$$\sigma_0^{(\ell)} = r_\ell \cdot \cdots\cdots r_2 r_1 \cdot x_i Q + tH(m),$$

$$\sigma_k^{(\ell)} = r_\ell \cdots r_k \cdot x_i P \qquad \text{for } k \in \{1, \ldots, \ell\},$$

$$\sigma_{-k}^{(\ell)} = r_k \cdot P \qquad \text{for } k \in \{\ell, \ldots, 1\},$$

$$\sigma_\infty = tP.$$

Recall Section 9 where we decomposed the signature scheme into simple building blocks. We now revise the building blocks for the modified scheme before we continue with the re-signing process.

Building blocks revisited. As we have seen above the modified signature scheme slightly differs from the original one in the H-representation. Therefore we first state how adding an H works and then we explain the re-randomization process by introducing a new building block `RE-RANDOM` ∞ which explains the

re-randomization of the new element σ_∞.

Adding an H to the modified signature scheme. Recall Figure 11.1 and consider Figure 15.2, a modified level ℓ signature differs from the original one only in the first H. Thus, adding an H to a level $\ell > 0$ signature is not different from what we have explained in Section 9. Figure 15.3 shows how adding a trivial H to a modified level 0 signature valid for the public key X_A works. The result is a level 1 signature still valid for the public key X_A.

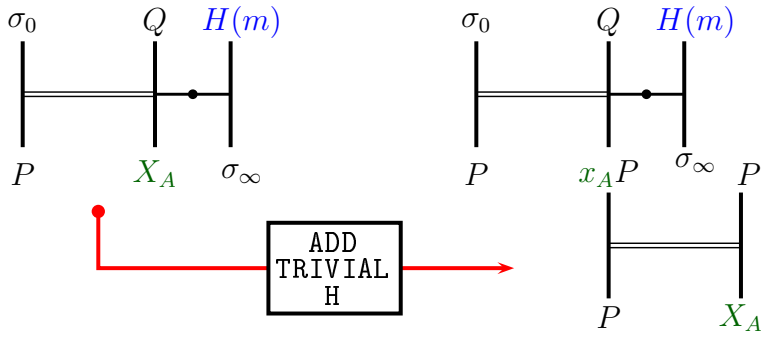


Figure 15.3: Adding a trivial H to a modified level 0 signature

Similarly Figure 15.4 shows how a re-signature H is added to a modified level 0 signature valid for the public key X_A with the re-signature key R_{AB} and the public key X_B of user B.

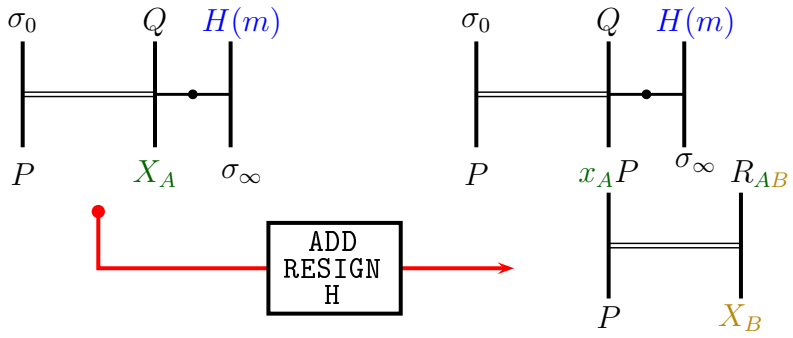


Figure 15.4: Adding a re-signature H to a modified level 0 signature

Re-randomization of the modified signature scheme. We start with introducing a new building block RE-RANDOM ∞ . This building block, on input a modified level ℓ signature

$$\sigma^{(\ell)} = (\sigma_0, \sigma_1, \ldots, \sigma_\ell, \sigma_{-\ell}, \ldots, \sigma_{-1}, \sigma_\infty) \in \mathbf{G}^{2\ell+2}$$

first choses a random $r' \xleftarrow{\$} \mathbb{Z}_p^\times$ and it calculates $\sigma_0 \leftarrow \sigma_0 + r'H(m)$ and $\sigma_\infty \leftarrow \sigma_\infty + r'P$, as shown in Figure 15.5.

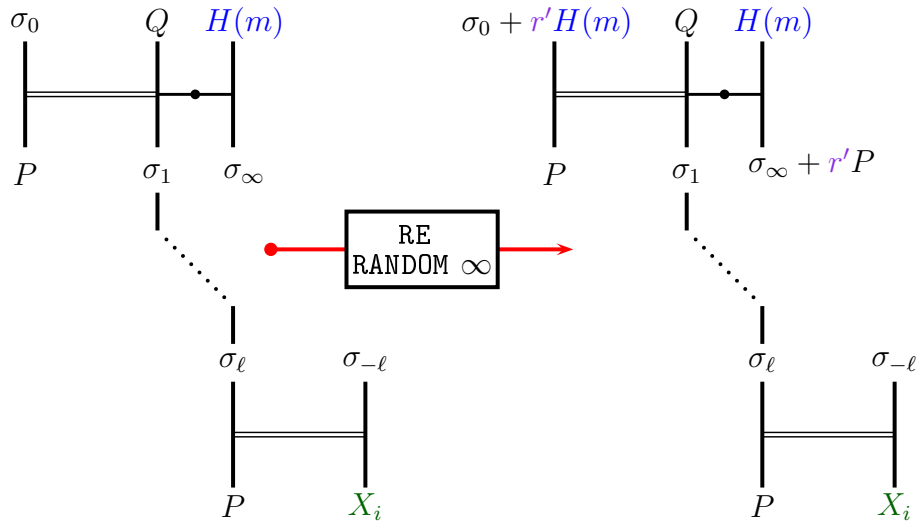


Figure 15.5: Re-randomizing σ_0 and σ_∞

The result is still a valid level ℓ signature in the modified scheme since

$$\begin{aligned} e(\sigma_0 + r'H(m), P) = & \\ &= e(\sigma_0, P) \cdot e(r'H(m), P) \\ &= e(x_i Q + tH(m), P) \cdot e(r'H(m), P) \\ &= e(x_i Q, P) \cdot e(tH(m), P) \cdot e(r'H(m), P) \\ &= e(Q, X_i) \cdot e(H(m), t + rP) \\ &= e(Q, X_i) \cdot e(H(m), \sigma_\infty). \end{aligned}$$

Randomizing at height i. In the modified scheme the building block RE-RANDOM i operates slightly different then in the original scheme. This building block, on input a modified level ℓ signature

$$\sigma^{(\ell)} = (\sigma_0, \sigma_1, \ldots, \sigma_\ell, \sigma_{-\ell}, \ldots, \sigma_{-1}, \sigma_\infty) \in \mathbf{G}^{2\ell+2}$$

first choses a random $t_i \xleftarrow{\$} \mathbb{Z}_p^\times$ and multiplies $\sigma_i, \sigma_{i-1}, \ldots, \sigma_0$ and σ_{-i} and also σ_∞ with it as shown in Figure 15.6. The result is a valid level ℓ signature in

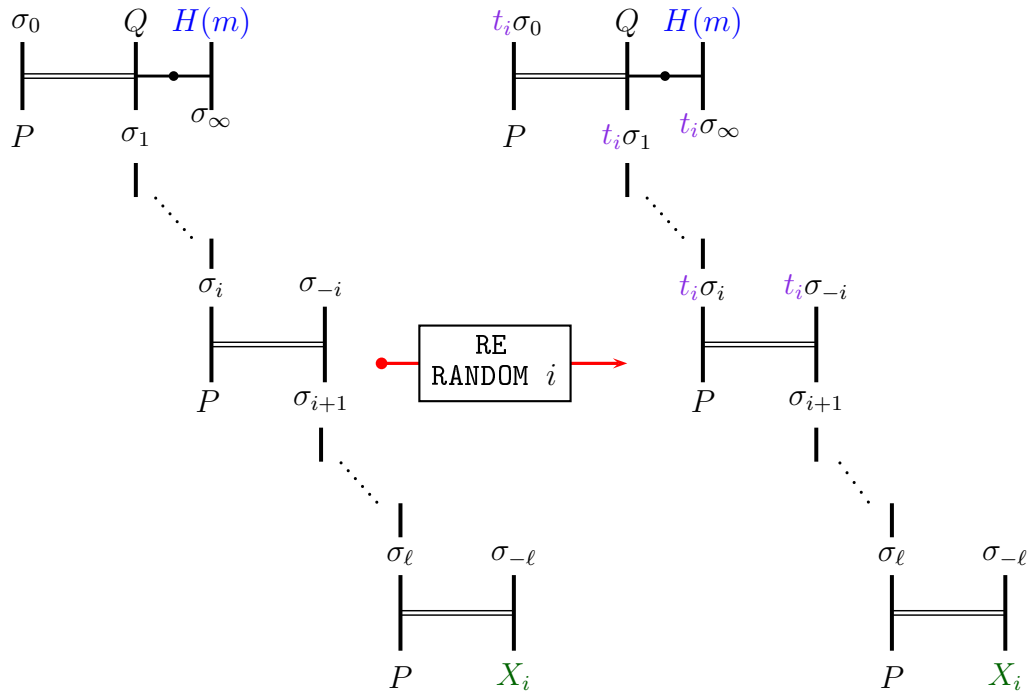


Figure 15.6: Re-randomizing $\sigma^{(\ell)}$ at height i

the modified scheme.

The full re-randomization of a level ℓ signature. Recall that for the original signature scheme we noted that RE-RANDOM $= \prod_{i=1}^{\ell}$ RE-RANDOM i . Here in the modified signature scheme we have RE-RANDOM $= \prod_{i=1}^{\ell}$ RE-RANDOM i + RE-RANDOM ∞ . This means that for a modified level ℓ signature, *first* all the RE-RANDOM i for $i \in \{1, \ldots, \ell\}$ are used and then RE-RANDOM ∞ is used *afterwards*, the result is shown in Figure 15.7.

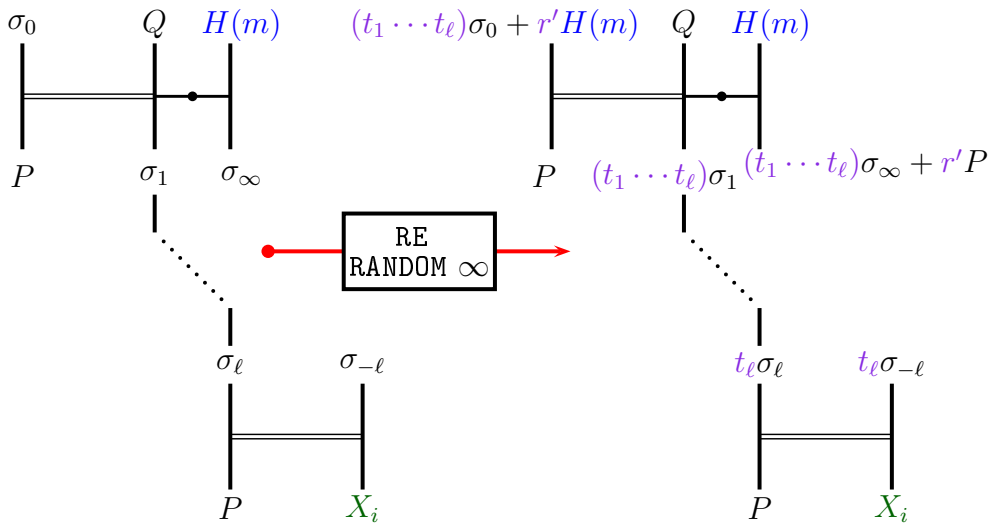


Figure 15.7: Re-randomizing $\sigma^{(\ell)}$ completely

Now we continue with the description of the modified signature scheme with the re-signature algorithm.

Re-Sign(cp, $m, \ell - 1, \sigma^{(\ell-1)}, R_{ij}, X_i, X_j$**):** On input a message $m \in \{0,1\}^n$, the re-signing key $R_{ij} = \frac{x_i}{x_j}P$ and a level $\ell - 1$ signature

$$\sigma^{(\ell-1)}(m) = \left(\sigma_0^{(\ell-1)}, \sigma_1^{(\ell-1)}, \sigma_2^{(\ell-1)}, \ldots, \sigma_{\ell-1}^{(\ell-1)}, \sigma_{-\ell+1}^{(\ell-1)}, \ldots, \sigma_{-1}^{(\ell-1)}, \sigma_\infty^{(\ell-1)}\right)$$

on m and the public keys X_i, X_j this algorithm first appends the re-signature H and then re-randomizes the result. During the computation the algorithm chooses $\ell + 1$ random coefficients $r', t_1, t_2, \ldots, t_\ell \xleftarrow{\$} \mathbb{Z}_p^\times$ and translates $\sigma^{(\ell-1)}$ into a level ℓ signature valid for the public key X_j by computing and outputting

$$\sigma_0^{(\ell)} = t_\ell \cdot \cdots\cdots t_2 t_1 \cdot \sigma_0^{(\ell-1)} + r' H(m),$$

$$\sigma_k^{(\ell)} = t_\ell \cdots t_k \cdot \sigma_k^{(\ell-1)} \qquad \text{for } k \in \{1, \ldots, \ell-1\},$$

$$\sigma_\ell^{(\ell)} = t_\ell X_i,$$

$$\sigma_{-\ell}^{(\ell)} = t_\ell R_{ij},$$

$$\sigma_{-k}^{(\ell)} = t_k \cdot \sigma_{-k}^{(\ell-1)} \qquad \text{for } k \in \{\ell, \ldots, 1-1\},$$

$$\sigma_\infty^{(\ell+1)} = t_\ell \cdot \cdots t_2 t_1 \cdot \sigma_\infty^{(\ell-1)} + r' P.$$

Since we know that the input signature was

$$\begin{aligned}
\sigma^{(\ell-1)}(m) &= \left(\sigma_0^{(\ell-1)}, \sigma_1^{(\ell-1)}, \sigma_2^{(\ell-1)}, \ldots, \sigma_{\ell-1}^{(\ell-1)}, \sigma_{-\ell+1}^{(\ell-1)}, \ldots, \sigma_{-1}^{(\ell-1)}, \sigma_\infty^{(\ell-1)}\right) \\
&= (r_{\ell-1} \cdot \cdots\cdots r_1 x_i Q + tH(m), \\
&\quad r_{\ell-1} \cdot \cdots\cdots r_1 x_i P, \\
&\quad r_{\ell-1} \cdots r_2 x_i P, \\
&\quad \vdots \\
&\quad r_{\ell-1} x_i P, \\
&\quad r_{\ell-1} P, r_{\ell-2} \ldots, r_1 P, \\
&\quad \tilde{t} P) \in \mathbf{G}^{2\ell},
\end{aligned}$$

setting $\overline{r}_\ell = t_\ell \frac{x_i}{x_j}$, $\overline{t} = t_1 \cdots t_\ell + r'$ and $\overline{r}_k = t_k r_k$ for $k \in \{1, \ldots, \ell\}$ gives us

$$\begin{aligned}
\sigma^{(\ell)}(m) &= (\sigma_0, \sigma_1, \ldots, \sigma_\ell, \sigma_{-\ell-1}, \ldots, \sigma_{-1}, \sigma_\infty) \\
&= (\ \overline{r}_\ell \ldots\ldots\ldots \overline{r}_1 x_j Q + \overline{t} H(m), \\
&\quad \overline{r}_\ell \ldots\ldots \overline{r}_1 x_j P, \\
&\quad \overline{r}_\ell \ldots \overline{r}_2 x_j P, \\
&\quad \vdots \\
&\quad \overline{r}_\ell x_j P, \\
&\quad \overline{r}_\ell P, \overline{r}_{\ell-1} P, \ldots, \overline{r}_1 P, \\
&\quad \overline{t} P) \in \mathbf{G}^{2\ell+2}.
\end{aligned}$$

This is a modified level ℓ signature valid for the public key X_j. Notice that in this scheme we have one more element at each level and this is slightly different than the original one but this modification will allow us to prove that the modified signature scheme is also secure in the standard model.

16. Proof of security in the standard model

The proof of security in the standard model is almost the same as it is in the random oracle model except that the hashing oracle is removed and replaced by a hash function.

THEOREM 16.1. *The modified proxy re-signature scheme with L levels and N users is secure under the ℓ-flexDH assumption. More precisely, given an attacker $\mathcal{A}$ to Game 13.1 with advantage ε we can construct an algorithm $\mathcal{B}$ that solves an ℓ-flexDH instance for given $P, aP, bP \in \mathbf{G}$ with*

$$Pr[\mathcal{B} \text{ is successful }] \geq \frac{q_{sk} \cdot q_{rk} \cdot (N - |I^*|)^3}{N^{|I^*|+3} \cdot 8(q_s + q_{rs})(n+1)} \cdot \varepsilon.$$

where $a, b \in \mathbb{Z}_p^\times$, n is the length of the messages to be signed, q_{sk} is the number of private key queries, q_{rk} is the number of re-signature key queries, q_s is the number of signature queries and q_{rs} is the number of re-signature queries made by $\mathcal{A}$.

PROOF. As in random oracle model proof, we construct an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ as a blackbox and simulates its oracles $\mathcal{O}_{\text{Skey}}$, $\mathcal{O}_{\text{ReKey}}$, $\mathcal{O}_{\text{Sign}}$ and $\mathcal{O}_{\text{ReSign}}$. As before $\mathcal{B}$ keeps track of the queries to these oracles in the Q-list to construct the query graph G_Q later. First $\mathcal{B}$ prepares the common public parameters as follows.

Prepare setup.

- Set $Q = bP$.
- Choose an integer $\tau \ll p$.
- Choose two random vectors $\vec{\omega} = (\omega', \omega_1, \ldots, \omega_n) \xleftarrow{\$} \mathbb{Z}_\tau^{n+1}$ and $\vec{z} = (z', z_1, \ldots, z_n) \xleftarrow{\$} \mathbb{Z}_p^{n+1}$, and a random integer $\kappa \xleftarrow{\$} \{0, \ldots, n\}$. Then define a vector $\vec{U} = (U', U_1, \ldots, U_n)$ as $U' = (\omega' - \kappa\tau)Q + z'P$ and $U_i = \omega_i Q + z_i P$ for $i \in \{0, \ldots, n\}$.

- For any message $m = m_1 \dots m_n \in \{0,1\}^n$ the hash value is defined as $H(m) = U' + \sum_{i=1}^{n} m_i U_i$. We also define two auxiliary functions $J(m)\colon \{0,1\}^n \to \mathbb{Z}$ and $K(m)\colon \{0,1\}^n \to \mathbb{Z}_p$ by

  - $J(m) = \omega' + \sum_{i=1}^{n} m_i \omega_i - \kappa\tau$,
  - $K(m) = z' + \sum_{i=1}^{n} m_i z_i$,

  so that $H(m) = J(m)Q + K(m)P$.

At the end the simulator will be successful when $\mathcal{A}$ comes up with a forged signature σ^* on a message m^* for which $J(m^*) \equiv 0 \mod p$ and for all other messages $m \neq m^*$ queried by $\mathcal{A}$, $J(m) \not\equiv 0 \mod p$. In fact here we assume $|J(\cdot)| \leq \tau(n+1) \ll p$ such that $J(m^*) \equiv 0 \mod p$ happens with non-negligible probability.

The attacker $\mathcal{A}$ is now being challenged with the system parameters $(P, Q, \vec{U})$. $\mathcal{B}$ answers the oracle calls of $\mathcal{A}$ as follows:

Public keys: As in the previous proof, $\mathcal{B}$ initially guesses a set of users $I^* \subset \{0, \dots, N-1\}$ and sets the public keys of users $i \in I^*$ as $X_i = z_i a P$ for some $z_i \xleftarrow{R} \mathbb{Z}_p^\times$ and all other public keys as $X_i = x_i P$, for some $x_i \xleftarrow{R} \mathbb{Z}_p^\times$ and makes them available to $\mathcal{A}$.

$\mathcal{B}$ answers the the oracle calls of $\mathcal{A}$ as follows.

Private key queries: When $\mathcal{A}$ asks for the secret key of user i, $\mathcal{B}$ does the following:

ALGORITHM $\mathcal{O}_{\text{Skey}}$.

Input: A user $i \in \{0, \dots, N-1\}$.
Output: The secret key x_i of user i or $\mathcal{B}$ aborts.

1. If $i \in I^*$ then $\mathcal{B}$ aborts.
2. Add $[\mathcal{O}_{\text{Skey}}, i]$ to Q-list.
3. Return x_i.

Re-signature key queries: When $\mathcal{A}$ queries $\mathcal{O}_{\text{ReKey}}$ for re-signature keys, $\mathcal{B}$ does the following:

ALGORITHM $\mathcal{O}_{\text{ReKey}}$.
Input: Two users $i, j \in \{0, \ldots, N-1\}$.
Output: The corresponding re-signature key R_{ij} or $\mathcal{B}$ aborts.

1. If $i \notin I^*$ then
2. If $j \in I^*$ then $\mathcal{B}$ aborts.
3. Else $R_{ij} \leftarrow \frac{x_i}{x_j} P$.
4. Else
5. If $j \in I^*$ then $R_{ij} \leftarrow \frac{z_i a}{z_j a} P = \frac{z_i}{z_j} P$.
6. Else $R_{ij} \leftarrow \frac{z_i a}{x_j} P$.
7. Add $[\mathcal{O}_{\text{ReKey}}, (i, j)]$ to Q-list.
8. Return R_{ij}.

Signature queries: When $\mathcal{A}$ asks for a signature of user i on m, $\mathcal{B}$ does the following:

ALGORITHM $\mathcal{O}_{\text{Sign}}$.
Input: A message $m \in \{0,1\}^n$, user i, a desired level ℓ.
Output: A modified level ℓ signature σ on m valid for X_i or $\mathcal{B}$ aborts.

1. If $J(m) \equiv 0 \mod p$ then $\mathcal{B}$ aborts.
2. Choose $t \xleftarrow{\$} \mathbb{Z}_p^\times$.
3. If $i \notin I^*$ then
4. $\sigma_0 \leftarrow (x_i Q + t \cdot H(m))$.
5. $\sigma_\infty \leftarrow tP$
6. Else
7. $\sigma_0 \leftarrow \left(-\frac{K(m)}{J(m)} X_i + t \cdot H(m)\right)$.
8. $\sigma_\infty \leftarrow \left(-\frac{1}{J(m)} X_i + t \cdot P\right)$.
9. Add $[i, m]$ to Q-list.
10. $\sigma \leftarrow (\sigma_0, \sigma_\infty)$
11. If $\ell > 0$ then
12. For $j = 1, \ldots, \ell$ do
13. $\sigma \leftarrow$ `ADD TRIVIAL H` $\blacktriangleleft \sigma$.
14. $\sigma \leftarrow$ `RE-RANDOM` $\blacktriangleleft \sigma$.
15. Return σ.

Observe that the elements of a level 0 signature generated on behalf of users $i \in I^*$ in steps 7 and 8 have the correct distribution since setting $t = \overline{t} + \frac{z_i a}{J(m)}$ for $\sigma = (\sigma_0, \sigma_\infty)$ yields

$$\begin{aligned}
\sigma_0 &= -\frac{K(m)}{J(m)} X_i + t \cdot H(m) \\
&= -\frac{K(m)}{J(m)} X_i + \overline{t} H(m) + \frac{z_i a}{J(m)} H(m) \\
&= -\frac{K(m)}{J(m)} X_i + \overline{t} H(m) + \frac{z_i a}{J(m)} (J(m)Q + K(m)P) \\
&= -\frac{K(m)}{J(m)} X_i + \overline{t} H(m) + z_i a Q + \frac{z_i a K(m)}{J(m)} P \\
&= \overline{t} H(m) + z_i a Q - \frac{K(m)}{J(m)} z_i a P + \frac{K(m)}{J(m)} z_i a P \\
&= z_i a Q + \overline{t} \cdot H(m)
\end{aligned}$$

and

$$\begin{aligned}
\sigma_\infty &= -\frac{1}{J(m)} X_i + \overline{t} P + \frac{z_i a}{J(m)} P \\
&= \overline{t} P - \frac{1}{J(m)} z_i a P + \frac{z_i a}{J(m)} P \\
&= \overline{t} P.
\end{aligned}$$

Note that the building blocks `ADD TRIVIAL H` and `RE-RANDOM` used here are the ones introduced for the modified scheme.

Re-signing queries: When $\mathcal{A}$ asks for a re-signature of the valid $\ell - 1$ signature $\sigma^{(\ell-1)}$ from user i to j on m, as in the random oracle model, $\mathcal{B}$ ignores this and uses $\mathcal{O}_{\text{Sign}}$ to create a level ℓ signature on m valid for user j. $\mathcal{B}$ executes

ALGORITHM $\mathcal{O}_{\text{ReSign}}$.

Input: A modified level $\ell - 1$ signature $\sigma^{(\ell-1)}$ valid for the public key X_i, two public keys X_i and X_j of users $i, j \in \{0, \ldots, N-1\}$ and a message $m \in \{0,1\}^n$.

Output: A modified level ℓ signature $\sigma^{(\ell)}$ on m valid for the public key X_j or $\mathcal{B}$ aborts.

1. $\sigma^{(\ell)} \leftarrow \mathcal{O}_{\text{Sign}}(m, j, \ell)$.

2. Add $[\mathcal{O}_{\text{ReSign}}, (i, j, m)]$ to Q-list.
3. Return $\sigma^{(\ell)}$.

As before, note that the call of algorithm $\mathcal{O}_{\text{Sign}}$ in step 1 can cause $\mathcal{B}$ to abort.

Final output: Finally when $\mathcal{A}$ outputs a message signature pair (m^*, σ^*) where $\sigma^* = (\sigma_0^*, \ldots, \sigma_\ell^*, \sigma_{-\ell}^*, \ldots, \sigma_{-1}^*, \sigma_\infty^*)$ is a valid level ℓ signature on m^* on behalf of user i^*. If initially $\mathcal{B}$ guessed I^* correctly and did not have to abort before, he does the following

FINALIZE.

Input: A message m^*, a modified level ℓ signature σ^* valid for the public key X_{i^*} on m^* and the general query list Q-list.

Output: An ℓ-flexDH instance $(abD_\ell, aD_\ell, \ldots, aD_1, C_\ell, \ldots, C_1)$ or $\mathcal{B}$ aborts.

1. Create the query graph $G_Q \leftarrow$ Query-graph$(i^*, m^*, Q\text{-list})$.
2. If there is a path from $[0]$ to $[i^*, m^*]$ in G_Q then $\mathcal{B}$ aborts.
3. If $J(m^*) \not\equiv 0 \mod p$ then $\mathcal{B}$ aborts.
4. Find the path π from $[i', m^*]$ to $[i^*, m^*]$ with length $|\pi| = k$.
5. Determine the order of users $\pi_0 = i', \ldots, \pi_k = i^*$ on the path π.
6. Calculate the elements $\frac{1}{z_{\pi_0}} = \frac{1}{z_{i'}}, \frac{1}{z_{\pi_1}}, \ldots, \frac{1}{z_{\pi_k}} = \frac{1}{z_{i^*}}$.
7. Return $\Bigg(\left(\frac{1}{z_{\pi_0}}\right)(\sigma_0^* - K(m^*)\sigma_\infty^*), (\frac{1}{z_{\pi_0}})\sigma_1^*, \ldots, (\frac{1}{z_{\pi_0}})\sigma_{\ell-k}^*, \left(\frac{1}{z_{\pi_1}}\right)\sigma_{\ell-k+1}^*,$
$\left(\frac{1}{z_{\pi_2}}\right)\sigma_{\ell-k+2}^*, \ldots, \left(\frac{1}{z_{\pi_k}}\right)\sigma_\ell^*,$
$\left(\frac{z_{\pi_k}}{z_{\pi_{k-1}}}\right)\sigma_{-\ell}^*, \left(\frac{z_{\pi_{k-1}}}{z_{\pi_{k-2}}}\right)\sigma_{-\ell+1}^*, \left(\frac{z_{\pi_{k-2}}}{z_{\pi_{k-3}}}\right)\sigma_{-\ell+2}^*, \ldots, \left(\frac{z_{\pi_1}}{z_{\pi_0}}\right)\sigma_{-\ell+k}^*,$
$\sigma_{-\ell+k+1}^*, \ldots, \sigma_{-2}^*, \sigma_{-1}^* \Bigg)$.

After renaming the coefficients accordingly, we know that

$$\sigma_0^* = r_\ell \cdots r_1 z_{i'} aQ + tK(m^*)P \quad \text{and} \quad \sigma_\infty^* = tP.$$

Thus, we have

$$\begin{aligned}\frac{1}{z_{\pi_0}}(\sigma_0^* - K(m^*)\sigma_\infty^*) &= \frac{1}{z_{i'}}(r_\ell \cdots r_1) z_{i'} aQ + z_{i'} tK(m^*)P - z_{i'} K(m^*) tP \\ &= abr_\ell \cdots r_1 P.\end{aligned}$$

This means that

$$\left(\left(\frac{1}{z_{\pi_0}}\right)(\sigma_0^* - K(m^*)\sigma_\infty^*), (\frac{1}{z_{\pi_0}})\sigma_1^*, \ldots, (\frac{1}{z_{\pi_0}})\sigma_{\ell-k}^*, \left(\frac{1}{z_{\pi_1}}\right)\sigma_{\ell-k+1}^*,\right.$$
$$\left(\frac{1}{z_{\pi_2}}\right)\sigma_{\ell-k+2}^*, \ldots, \left(\frac{1}{z_{\pi_k}}\right)\sigma_\ell^*,$$

$$\left(\frac{z_{\pi_k}}{z_{\pi_{k-1}}}\right)\sigma_{-\ell}^*, \left(\frac{z_{\pi_{k-1}}}{z_{\pi_{k-2}}}\right)\sigma_{-\ell+1}^*, \left(\frac{z_{\pi_{k-2}}}{z_{\pi_{k-3}}}\right)\sigma_{-\ell+2}^*, \ldots, \left(\frac{z_{\pi_1}}{z_{\pi_0}}\right)\sigma_{-\ell+k}^*,$$

$$\left.\sigma_{-\ell+k+1}^*, \ldots, \sigma_{-2}^*, \sigma_{-1}^*\right)$$
$$= (abD_\ell, aD_\ell, \ldots, aD_1, C_\ell, \ldots, C_1),$$

with $D_i = r_i \cdots r_1 P$ and $C_i = r_i P$ is a valid ℓ-flexDH instance, since for all $j \in \{1, \ldots, \ell\}$ we have $\log_P D_j = \prod_{i=1}^{j} \log_P C_i$ and C_i is not the neutral element of the group $\mathbf{G}$.

The success probability of $\mathcal{B}$ As in the proof of security in the random oracle model the initial guess of I^* gives us the probability

$$\frac{1}{\binom{N}{|I^*|}} \geq \frac{1}{N^{|I^*|}}.$$

The probability of $\mathcal{B}$ not aborting for q_{sk} many secret key queries of $\mathcal{B}$ we have the probability

$$q_{sk}\frac{N - |I^*|}{N}.$$

Also, for q_{rk} many re-signature key queries of $\mathcal{A}$ we have the probability of $\mathcal{B}$ not aborting

$$q_{rk}\left(1 - \frac{N - |I^*|}{N} \cdot \frac{|I^*|}{N-1}\right) \geq q_{rk}\frac{(N - |I^*|)^2}{N^2}.$$

Now, following Waters (2005) we show that the probability of $J(m^*) \equiv 0 \mod p$ is

$$\Pr[J(m^*) \equiv 0 \mod p] \geq \frac{1}{8(q_s + q_{rs})(n+1)},$$

where n is the length of the messages to be signed. This yields the announced bound of $\mathcal{B}$'s advantage. To simplify the analysis we define another auxiliary function for an n-bit message $M = m_1, \ldots, m_n$:

$$F(M) = \begin{cases} 0, & \text{if } \omega' + \sum_{i=1}^{n} m_i \omega_i \equiv 0 \mod \tau, \\ 1, & \text{otherwise.} \end{cases}$$

Then the probability of $J(m^*) \equiv 0 \mod p$ is given by

$$Pr[J(m^*) \equiv 0 \mod p] = Pr\left[\left(\bigwedge_{i=1}^{q_s+q_{rs}} F(M_i) = 1\right) \wedge \omega' + \sum_{i=1}^{n} m_i^* \omega_i = \kappa\tau\right]$$

for $q_s + q_{rs}$ queries on n-bit messages $M_j = m_{j1}, \ldots, m_{jn}$, $j \in \{1, \ldots, q_s + q_{rs}\}$ and the challenge message $m^* = m_1 \ldots m_n$. Note here that $F(M) \neq 0$ implies that $J(M) \not\equiv 0 \mod p$ because of the initial assumption that $p \gg \tau(n+1)$. Note also that every re-signature query in fact triggers a signature query, as mentioned above. First we rewrite the probability from above as

$$\begin{aligned} Pr[J(m^*) \equiv 0 \mod p] &= Pr\left[\left(\bigwedge_{i=1}^{q_s+q_{rs}} F(M_i) = 1\right) \wedge \omega' + \sum_{i=1}^{n} m_i^* \omega_i = \kappa\tau\right] \\ &= Pr\left[\left(\bigwedge_{i=1}^{q_s+q_{rs}} F(M_i) = 1\right)\right] \cdot Pr\left[\omega' + \sum_{i=1}^{n} m_i^* \omega_i = \kappa\tau \,\middle|\, \bigwedge_{i=1}^{q_s+q_{rs}} F(M_i) = 1\right] \\ &\geq \left(1 - Pr\left[\bigvee_{i=1}^{q_s+q_{rs}} F(M_i) = 0\right]\right) \cdot Pr\left[\omega' + \sum_{i=1}^{n} m_i^* \omega_i = \kappa\tau \,\middle|\, \bigwedge_{i=1}^{q_s+q_{rs}} F(M_i) = 1\right]. \end{aligned} \tag{16.2}$$

Since we know that $Pr[F(M) = 0] = \frac{1}{\tau}$, we transform equation (16.2) into

$$\left(1 - \frac{q_s + q_{rs}}{\tau}\right) \cdot Pr\left[\omega' + \sum_{i=1}^{n} m_i^* \omega_i = \kappa\tau \,\middle|\, \bigwedge_{i=1}^{q_s+q_{rs}} F(M_i) = 1\right]. \tag{16.3}$$

We also know that $\kappa \xleftarrow{\$} \{0, \ldots, n\}$ which gives us the factor $\frac{1}{n+1}$ such that we can also use the auxiliary function $F(\cdot)$ for m^*. Thus, we change equation (16.3) into

$$\frac{1}{n+1} \cdot \left(1 - \frac{q_s + q_{rs}}{\tau}\right) \cdot Pr\left[F(m^*) = 0 \,\middle|\, \bigwedge_{i=1}^{q_s+q_{rs}} F(M_i) = 1\right]. \tag{16.4}$$

Now we use *Bayes' theorem* to transform the probability in equation (16.4) into

$$\frac{Pr\left[F(m^*)=0\right]}{Pr\left[\bigwedge_{i=1}^{q_s+q_{rs}} F(M_i)=1\right]} \cdot Pr\left[\bigwedge_{i=1}^{q_s+q_{rs}} F(M_i)=1 \middle| F(m^*)=0\right].$$

Here again we know that $Pr\left[F(M)=0\right]=\frac{1}{\tau}$ so we can estimate that

$$\frac{Pr\left[F(m^*)=0\right]}{Pr\left[\bigwedge_{i=1}^{q_s+q_{rs}} F(M_i)=1\right]} \text{ is at least } \frac{1}{\tau}.$$

Thus, with equation (16.4) we get

$$Pr[J(m^*) \equiv 0 \bmod p] \geq \frac{1}{n+1}\cdot\left(1-\frac{q_s+q_{rs}}{\tau}\right)\cdot\frac{1}{\tau}\cdot Pr\left[\bigwedge_{i=1}^{q_s+q_{rs}} F(M_i)=1 \middle| F(m^*)=0\right]$$

which is equal to

$$(16.5)\quad \frac{1}{n+1}\cdot\frac{1}{\tau}\cdot\left(1-\frac{q_s+q_{rs}}{\tau}\right)\cdot\left(1-Pr\left[\bigvee_{i=1}^{q_s+q_{rs}} F(M_i)=0 \middle| F(m^*)=0\right]\right).$$

We estimate that this equation is at least

$$(16.6)\quad \geq \frac{1}{n+1}\cdot\frac{1}{\tau}\cdot\left(1-\frac{q_s+q_{rs}}{\tau}\right)\cdot\left(1-\sum_{i=1}^{q_s+q_{rs}} Pr\left[F(M_i)=0 \middle| F(m^*)=0\right]\right).$$

Now we use that for any message pair M, M' the probabilities $F(M)=0$ and $F(M')=0$ are pairwise independent, since the sums $\omega' + \sum_{i=1}^{n} m_i\omega_i$ will differ at least in one random ω_i because $m_i \in \{0,1\}$. Therefore equation (16.6) is equal to

$$\begin{aligned}&\frac{1}{n+1}\cdot\frac{1}{\tau}\cdot\left(1-\frac{q_s+q_{rs}}{\tau}\right)\cdot\left(1-\frac{q_s+q_{rs}}{\tau}\right)\\ =&\frac{1}{n+1}\cdot\frac{1}{\tau}\cdot\left(1-\frac{q_s+q_{rs}}{\tau}\right)^2.\end{aligned}$$

Now we can optimize this by setting $\tau = 4(q_s+q_{rs})$ which gives us

$$\Pr[J(m^*) \equiv 0 \bmod p] \geq \frac{1}{8(q_s+q_{rs})(n+1)}.$$

Altogether we have the announced bound on the success probability of $\mathcal{B}$ as

$$Pr[\mathcal{B} \text{ is successful }] \geq \frac{q_{sk}\cdot q_{rk}\cdot(N-|I^*|)^3}{N^{|I^*|+3}\cdot 8(q_s+q_{rs})(n+1)}\cdot\varepsilon.$$

Artificial abort stage: Note that when τ is chosen in the beginning the number of signature queries and the number of re-signature queries $q_s + q_{rs}$ is not known to $\mathcal{B}$. Only after $\mathcal{A}$ outputs a message signature pair (m^*, σ^*), $\mathcal{B}$ knows the set of queried messages $\{M_1, \ldots, M_{q_s+q_{rs}}\}$, the forged message m^* and the value $q_s + q_{rs}$. This is corrected by $\mathcal{B}$ with an artificial abort stage before the final output. Now we explain this stage which happens between steps 3 and 4 of algorithm FINALIZE.

Assume that $\mathcal{B}$ aborts *before* this artificial abort stage for all sets of possible queries of $\mathcal{A}$ with almost the same probability $(1 - \zeta)$. Now we define a binary function $\alpha(\vec{w}, \mathcal{M}, m^*)$ for a set of simulation values $\vec{w} \in \mathbb{Z}_\tau^{n+1}$, $\mathcal{M} = \{M_1, \ldots, M_{q_s+q_{rs}}\}$ and m^* as

$$\alpha(\vec{w}, \mathcal{M}, m^*) = \begin{cases} 0, & \text{if } \left(\bigwedge_{i=1}^{q_s+q_{rs}} F(M_i) = 1\right) \wedge \omega' + \sum_{i=1}^{n} m_i^* \omega_i = \kappa\tau, \\ 1, & \text{otherwise.} \end{cases}$$

The function $\alpha(\vec{w}, \mathcal{M}, m^*)$ will evaluate to 0 if the set of queried messages $\mathcal{M}$ and the forged message m^* do not cause $\mathcal{B}$ to abort for the simulation values $\vec{w}$. We consider this probability as $Pr[(\alpha(\vec{w}, \mathcal{M}, m^*) = 0] = \eta$. At this stage, $\mathcal{B}$ collects with respect to ζ *enough* samples of the probabilities η by choosing a random $\vec{w}$ and evaluating $\alpha(\vec{w}, \mathcal{M}, m^*)$ to compute an estimated η'. Recall that we have $J(m^*) = \omega' + \sum_{i=1}^{n} m_i^* \omega_i - \kappa\tau \equiv 0 \mod p$ (step 3 of algorithm FINALIZE). Therefore this sampling does not require running $\mathcal{A}$ again, $\mathcal{B}$ just needs to find a right $\vec{\omega}$ for $\mathcal{M}$ and m^*. Then, if the estimated value η' is at least the probability ζ, ie. $\eta' \geq \zeta$, $\mathcal{B}$ aborts with probability $1 - \frac{\zeta}{\eta'}$. Here is $\zeta = \frac{1}{8(q_s+q_{rs})(n+1)}$ the lower bound of the probability of $\mathcal{B}$ not aborting at this stage, as we showed above. □

This unusual proof technique was adopted by many other publications and also from Libert & Vergnaud (2008a). For more details we refer to the original publication of Waters (2005). Note that the artificial abort stage can be very time consuming as mentioned in Bellare & Ristenpart (2009) which also shows how to eliminate this artificial abort stage and provides a more concrete estimation of the success probability.

Note that using the adversaries from the original security definition for this reduction gives us similar results as discussed in Section 14.1. We only get an additional factor $\frac{1}{8(n+1)}$ to the success probabilities of the adversaries $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and $\mathcal{A}_4$. This factor is the result of the instantiation of the hashing oracle with the hash function used in this reduction. Notice also that the factor

$$q_{sk} \cdot q_{rk} \cdot \frac{(N - |I^*|)^3}{N^{|I^*|+3}}$$

is the same as it is in the random oracle model. This means that the results achieved in Section 14.1 also apply here. Especially we require $|I^*| \in \mathrm{O}(1)$ to have a non-negligible success probability of $\mathcal{B}$. Further, we can also omit the factor related to the secret key queries of $\mathcal{A}$ as discussed before.

Part IV
Efficiency

In this section we will discuss some efficiency issues of the signature scheme. Considering the efficiency of the signature we see that the length of the signature and the growing number of random coefficients at each translation are the dominating factors. We begin with the discussion of the signature length and introduce a new problem called the *chain shortening problem* (CSP). The CSP will help us to get an idea about the implications of a shorter signature. We then discuss the number of coefficients used to build a level ℓ signature. We will observe what problems lesser or related coefficients may cause when they are used. Note that in this section we will only look at the basic form of the signature scheme and *not* the modified version which we defined for the standard model proof of security in the previous section.

17. Shortening the Signature

As mentioned before, the ReSign($\cdot$) algorithm increases the size of the signature by two elements with each translation. We recall that a level ℓ signature has the following elements:

$$\sigma_0^{(\ell)} = (r_\ell \cdots r_1) x_A H(m),$$

$$\begin{aligned}
\sigma_1^{(\ell)} &= (r_\ell \cdot \cdots \cdot \cdot r_1) x_A P, & \sigma_{-1}^{(\ell)} &= r_1 P,\\
\sigma_2^{(\ell)} &= (r_\ell \cdot \cdots \cdot r_2) x_A P, & \sigma_{-2}^{(\ell)} &= r_2 P,\\
\sigma_3^{(\ell)} &= (r_\ell \cdots r_3) x_A P, & \sigma_{-3}^{(\ell)} &= r_3 P,\\
&\vdots & &\vdots\\
\sigma_\ell^{(\ell)} &= r_\ell x_A P, & \sigma_{-\ell}^{(\ell)} &= r_\ell P.
\end{aligned}$$

Recall Part II where we introduced the H-representation to show the connection between these elements. Now, someone might claim to have an algorithm which can merge some of these H-s together so that a shorter signature might be achieved. We take this into consideration by assuming the existence of a

blackbox algorithm (an oracle) that somehow shortens the signature. To introduce this consider Figure 17.1 where a blackbox $CS_{2\to1}$ shortens the level 2 signature to a level 1 signature.

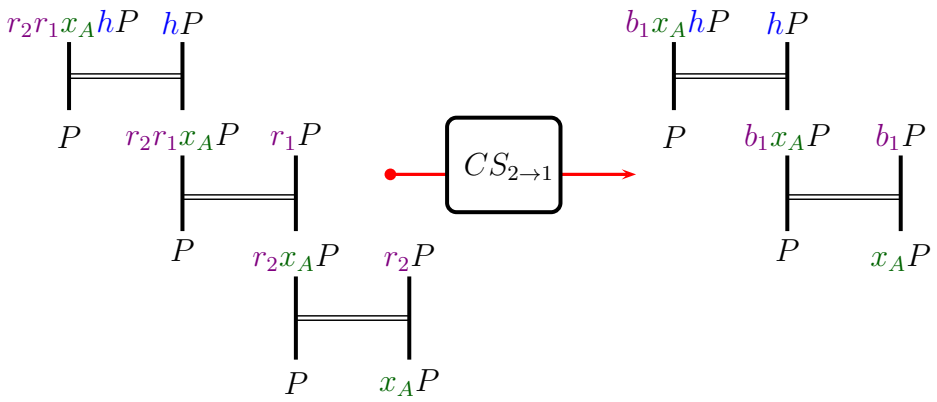


Figure 17.1: $CS_{2\to1}$

Here the *chain shortener* $CS_{2\to1}$ shortens the signature by one element by replacing $r_1, r_2 \in \mathbb{Z}_p^\times$ with a random $b_1 \xleftarrow{\$} \mathbb{Z}_p^\times$. This means that $CS_{2\to1}$ reduces the signature by one H as depicted in Figure 17.1. Note that $CS_{2\to1}$ is not allowed to change $X_A = x_AP$ and $H(m) = hP$, since the message m and the validation public key x_AP must stay the same after the shortening process. This is because after the shortening process we still want to have a signature on the same message m valid for the same public key X_A. Therefore x_AhP also cannot be changed during the shortening process (the green and blue elements). For now we will consider signatures of higher levels and come back to level 1 signatures later.

Now with the assumption that the *chain shortener* $CS_{2\to1}$ exists as a blackbox, we attempt to use it for shortening a level 3 signature as shown in Figure 17.2.

At first we decouple the lower H from the signature and use $CS_{2\to1}$. Then we recouple the H with the elements r_3P and x_AP to obtain a valid level 2 signature as shown in Figure 17.3. This implies the existence of the $CS_{3\to2}$ which shortens the level 3 signature to a level 2 signature.

Thus we get a valid level 2 signature for X_A as

$$\sigma^{(2)} = \quad (b_1P, \quad r_3P, \quad r_3x_AP, \quad b_1r_3x_AP, \quad b_1r_3x_AH(m)) \,.$$

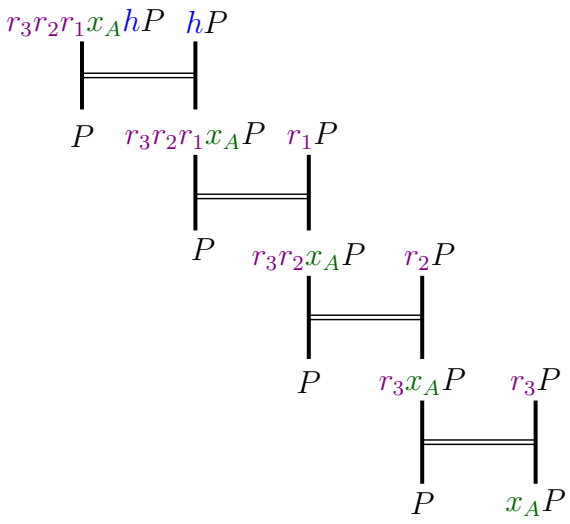


Figure 17.2: A level 3 signature

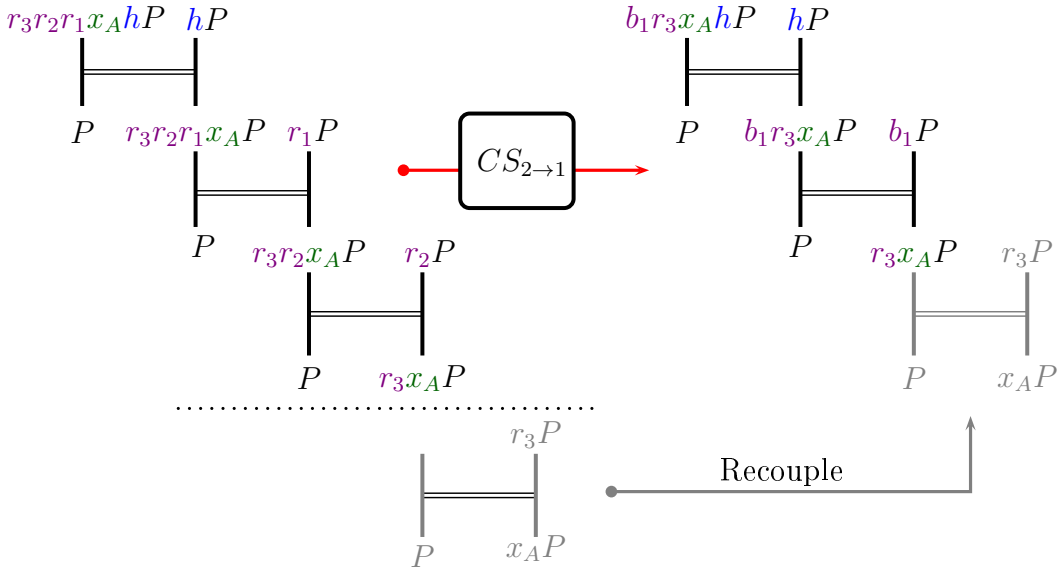


Figure 17.3: Using $CS_{2\to1}$ on a level 3 signature

Now, we can use $CS_{2\to1}$ again as shown in Figure 17.4, which gives us a level 1 signature valid for X_A. This implies the existence of the $CS_{3\to1}$.

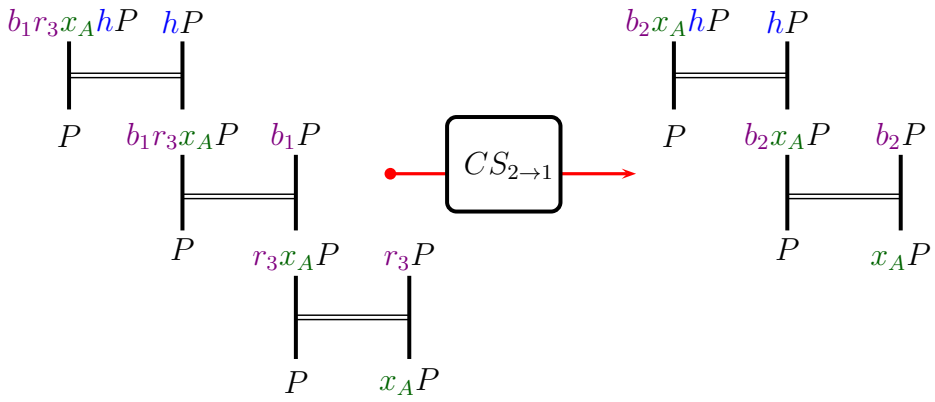


Figure 17.4: Shortening the signature again

Recapitulating the whole process, we first decouple the lower H from the level 3 signature and use $CS_{2\to1}$ to shorten it. Then we add the decoupled H to the result and obtain a level 2 signature. Using $CS_{2\to1}$ on that gives us a level 1 signature. Therefore we note that the chain shorteners $CS_{3\to2}$ and $CS_{3\to1}$ can be achieved from $CS_{2\to1}$ as

- $CS_{3\to2}$**:** Decouple lower H, use $CS_{2\to1}$, recouple the decoupled H.
- $CS_{3\to1}$**:** Combine $CS_{3\to2}$ and $CS_{2\to1}$.

Note that the existence of $CS_{3\to1}$ also implies the existence of $CS_{2\to1}$ since, we can easily lengthen the signature with our building block `ADD TRIVIAL H` (see Section 9) and use the $CS_{3\to1}$ afterwards.

Level 4 and higher. Similarly as above we can recursively construct $CS_{4\to3}$, $CS_{4\to2}$ and $CS_{4\to1}$ for level 4 signatures from a $CS_{2\to1}$.

- $CS_{4\to3}$**:** Decouple lower H, use $CS_{3\to2}$, recouple the decoupled H.
- $CS_{4\to2}$**:** Combine $CS_{4\to3}$ and $CS_{3\to2}$.
- $CS_{4\to1}$**:** Combine $CS_{4\to2}$ and $CS_{2\to1}$.

Again the implication $CS_{4\to1} \Rightarrow CS_{3\to1}$ is trivial since we can lengthen the signature easily before using $CS_{4\to1}$.

Analogously we can construct chain shorteners $CS_{\ell\to i}$ where $1 < i < \ell$. It even turns out that the chain shorteners form an implication hierarchy. This will give us an understanding about the implications of shortening a signature. But before formally writing down the results we return to the level 1 signatures.

The chain shortener for level 1 signatures. Now we introduce a chain shortener for a level 1 signature. A level 1 signature is given by

$$\sigma^{(1)} = (r_1 P, \quad r_1 x_A P, \quad r_1 x_A H(m)) .$$

We assume the existence of a chain shortener $CS_{1\to 0}$ which shortens this level 1 signature to a level 0 signature as shown in Figure 17.5.

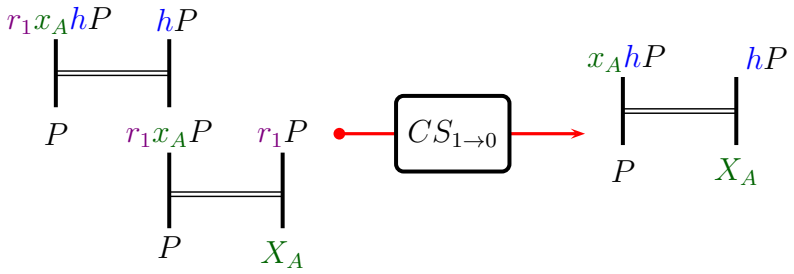


Figure 17.5: $CS_{1\to 0}$

Here again we can recursively build a $CS_{i\to 0}$ for $2 \leq i \leq \ell$ as we did before. For example we can build a $CS_{2\to 0}$ from the $CS_{1\to 0}$ as:

1. Decouple the lower H from the level 2 signature,
2. use $CS_{1\to 0}$ to shorten it,
3. add the decoupled H,
4. use $CS_{1\to 0}$ again to obtain a valid level 0 signature.

We then can use $CS_{2\to 0}$ similarly to build a $CS_{3\to 0}$ and that to build a $CS_{4\to 0}$ and so on.

The existence of $CS_{1\to 0}$ also implies the existence of $CS_{2\to 1}$ as:

1. Decouple the lower H from the level 2 signature,
2. use $CS_{1\to 0}$ to shorten it,

3. add the decoupled H and obtain a valid level 1 signature.

The existence of a $CS_{1\to 0}$ also enables us to calculate $a^{-1}P$ for given P, aP. We use $CS_{1\to 0}$ as as shown in Figure 17.6 to obtain $a^{-1}P$. We call this an inverter

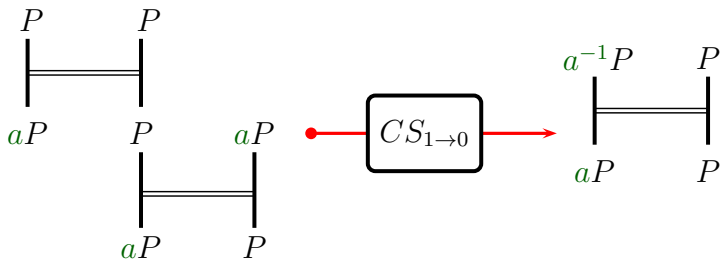


Figure 17.6: Obtaining $a^{-1}P$ with $CS_{1\to 0}$

and note it as `INV` shown in Figure 17.7.

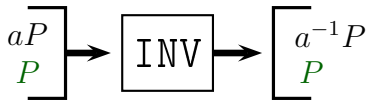


Figure 17.7: The inverter `INV`

17.1. The hierarchy of the chain shortening problem. Now we formally write down our results and analyze the implications of these results. For chain shorteners we can state:

THEOREM 17.1. *$\exists\ CS_{i'\to j} \Rightarrow \exists\ CS_{i\to j}$ for $j < i < i'$. This means that if we can shorten a level i' signature to a level j signature we can also shorten a level i signature to a level j signature where $i' > i$.*

PROOF. We can lengthen the level i signature with `ADD TRIVIAL H` as much as necessary until we have a level i' signature and then use the $CS_{i'\to j}$. □

THEOREM 17.2. $\exists\, CS_{i\to j} \Rightarrow \exists\, CS_{i\to j'}$ *for* $j < j' < i$. *This means that if we can shorten a level* i *signature to a level* j *signature we can also shorten it to a level* j' *signature, where* $j < j'$.

PROOF. Since lengthening the signature is easy we can use $CS_{i\to j}$ to shorten the level i signature to a level j signature and then use ADD TRIVIAL H as often as necessary and re-randomize it with RE-RANDOM to get a level j' signature. □

THEOREM 17.3. $\exists\, CS_{i\to j} \Rightarrow \exists\, CS_{i'\to j}$ *for* $j < i < i'$. *This means that if we can shorten a level* i *signature to a level* j *signature we can also shorten a level* i' *signature to a level* j *signature, where* $i < i'$.

PROOF. To shorten a level i' signature to a level j signature with $CS_{i\to j}$ where $i' > i$, we first have to decouple the lower $i' - i$ H s from the level i' signature and use $CS_{i\to j}$ to shorten it. Then add $i - j$ H s from the top of the decoupled ones and use $CS_{i\to j}$ again and continue this process until no decoupled H s are left. This is the same process we did above for $CS_{2\to 1}$ and $CS_{3\to 1}$. Note that in the case where $(i - j) \nmid (i' - i)$ we still can use ADD TRIVIAL H to lengthen the signature accordingly. □

THEOREM 17.4. $\exists\, CS_{i\to j} \wedge \exists\, CS_{j\to k} \Rightarrow \exists\, CS_{i\to k}$ *for* $k < j < i$. *A* $CS_{i\to k}$ *can easily be constructed from the combination of* $CS_{i\to j}$ *and* $CS_{i\to k}$.

PROOF. We first use the $CS_{i\to j}$ on a level i signature to get a level j signature then we use $CS_{j\to k}$ to obtain a level k signature. □

Consider Figure 17.8 which shows the implication hierarchy of the chain shorteners. Theorem 17.1 and Theorem 17.3 note the implications in the vertical direction, Theorem 17.2 notes the implications from right to left. The diagonal implications result from the combination of Theorem 17.3 and Theorem 17.2.

Now we prove the equivalence

$$\exists\, \mathtt{CS}_{1\to 0} \iff \mathtt{CDH}$$

with a series of lemmas.

LEMMA 17.5. $\exists\, CS_{1\to 0} \Rightarrow \exists$ *INV*. *The existence of* $CS_{1\to 0}$ *implies the existence of* *INV*.

PROOF. Use $CS_{1\to 0}$ with the basepoint and aP as shown in Figure 17.6. □

Similar to INV we define a squarer SQR which returns a^2P for given P, aP as in Figure 17.9.

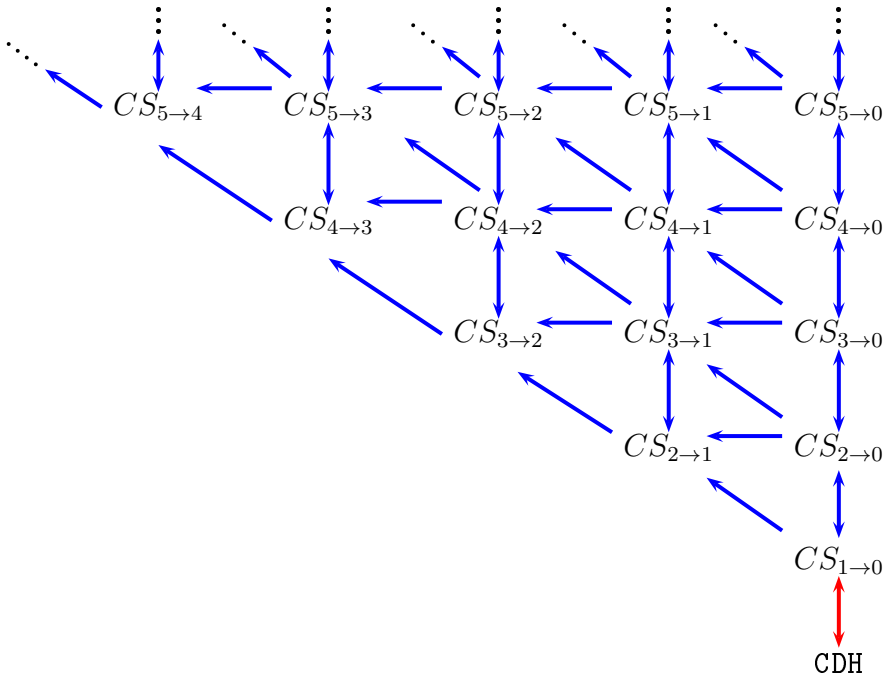


Figure 17.8: The Hierarchy of the chain shorteners

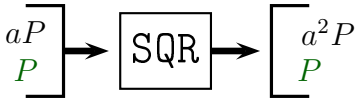


Figure 17.9: The squarer SQR

LEMMA 17.6. $\exists$ *INV* $\Rightarrow \exists$ *SQR*. *The existence of* *INV* *as in Figure 17.7 implies the existence of* *SQR*.

PROOF. Use the INV with the basepoint and aP to aquire $[a^2P]$ as follows

$$[aP, \quad P] \quad \xrightarrow{\text{INV}} \quad [a^2P].$$

□

LEMMA 17.7. *Assume that halving is easy and an efficient bilinear map $e(\cdot,\cdot)$ exists. Then $\exists$ [SQR] $\Leftrightarrow$ CDH. The existence of a squarer [SQR] as in Figure 17.9 is equivalent to solving the computational Diffie-Hellman problem (CDH).*

PROOF. $\exists$ SQR $\Rightarrow$ CDH.

Input: A base point P, two points aP and bP for $a, b \in \mathbb{Z}_p^\times$, a non-degenerate bilinear map $e(\cdot,\cdot)$, a squarer [SQR].
Output: The solution abP to the given CDH instance P, aP, bP.

1. Compute a^2P with [SQR].
2. Compute b^2P with [SQR].
3. Compute $(a+b)^2P$ with [SQR].
4. Compute $S \leftarrow (a+b)^2P - a^2P - b^2P = 2abP$.
5. Compute $\{c_1P, c_2P\} \leftarrow \frac{1}{2}S$.
6. Use $e(c_iP, P) = e(aP, bP)$ for $i \in \{1,2\}$ to determine the correct solution.
7. Return the correct solution $c_iP = abP$

Note that the addition and the halving operation are considered to be easy on elliptic curves. The latter is basically a square root operation which has two solutions. To determine the correct answer we use the bilinear map $e(\cdot,\cdot)$ which is also considered to be an easy operation.

Now we show CDH $\Rightarrow \exists$ [SQR]. This is trivial since the solution to the given CDH instance $[P, aP, aP]$ is a^2P. This means that if we have a blackbox [CDH] which returns abP for the input $[P, aP, bP]$ we can calculate a^2P with the input $[P, aP, aP]$. □

LEMMA 17.8. $\exists$ [CDH] $\Rightarrow \exists$ [INV]. *The existence of* [CDH] *implies the existence of an inverter* [INV] *as described in Figure 17.7.*

PROOF. Use [CDH] with input $[aP, P, P]$ to obtain $a^{-1}P$. □

LEMMA 17.9. $\exists$ [CDH] $\Rightarrow \exists\, CS_{1\to 0}$. *The existence of* [CDH] *implies the existence of the chain shortener $CS_{1\to 0}$.*

PROOF. Recall that a level 1 signature is given by

$$\big[P,\ x_AP,\ hP,\ r_1P,\ r_1x_AP,\ r_1x_AhP\big],$$

to obtain a level 0 signature from these we only need $x_A hP$. Thus using $\boxed{\texttt{CDH}}$ with input $[P, x_A P, hP]$ gives us a level 0 signature as:

$$\big[P,\ x_A P,\ hP,\ x_A hP\big].$$

□

The last result is very natural since the level 0 signature is a short signature (Boneh *et al.* 2004) which is based on the computational Diffie-Hellman assumption. Now we summarize the results from above:

$$\exists\ \mathtt{CS}_{1\to 0} \Rightarrow \exists\ \boxed{\texttt{INV}} \Rightarrow \exists\ \boxed{\texttt{SQR}} \Longleftrightarrow \exists\ \boxed{\texttt{CDH}}\ ,$$

$$\exists\ \boxed{\texttt{CDH}} \Rightarrow \exists\ \boxed{\texttt{INV}}\ ,$$

$$\exists\ \boxed{\texttt{CDH}} \Rightarrow \exists\ \mathtt{CS}_{1\to 0}\ .$$

Completing the hierarchy of the chain shorteners from Figure 17.8 we get:

$$\exists\ \boxed{\texttt{CDH}} \Longleftrightarrow \exists\ \mathtt{CS}_{1\to 0}\ .$$

Naturally, if we had a blackbox solving ℓ-flexDH instances we could also build a $CS_{i\to\ell}$ from that. For example, assume that a blackbox $\boxed{\texttt{1-flexDH}}$ exists which outputs $Q, aQ, abQ \in \mathbf{G}$ for input $P, aP, bP \in \mathbf{G}$. Using $\boxed{\texttt{1-flexDH}}$ with input $P, x_A P, hP$ would result in $Q, x_A Q, x_A hQ$, where hP is the hash value of the message and the public key $x_A P$. The result is basically a level 1 signature given by

$$(Q,\quad x_A Q,\quad x_A hQ),$$

considering that $Q = rP$ for some $r \xleftarrow{\$} \mathbb{Z}_p^\times$. Since we only need the public key $x_A P$ and the hash value hP we can shorten level ℓ signatures to level 1 signatures with $\boxed{\texttt{1-flexDH}}$. However, it is not clear how to relate $CS_{\ell\to 1}$ to $\boxed{\texttt{1-flexDH}}$ since a $CS_{\ell\to 1}$ generates a new 1-flexDH tuple from a given ℓ-flexDH tuple. Intuitively, every $CS_{i\to j}$ where $j \geq 1$ has some randomness that we cannot control. This prevents us also from relating these chain shorteners to more classical problems such as the CDH.

In the end, we observe that the existence of a $CS_{i\to 0}$ implies all other $CS_{i\to j}$ for $0 \leq j < i$ and this is also equivalent to solving CDH. For all other $CS_{i\to j}$ where $j \geq 1$ we can build other chain shorteners but reaching to any $CS_{i\to 0}$ seems not possible.

Even so shortening the signature seems hard to achieve. In conclusion, achieving logarithmic or sublinear or even constant length signatures seems out of reach at the moment (Libert & Vergnaud 2008a).

18. Usage of Random Coefficients

In this section we will analyze the usage of random coefficients. The main question we will try to answer is "*How much randomness is needed?*". Recall that the signature scheme is *not* SEUF as noted in Part III, since it can be publicly re-randomized with the building block RE-RANDOM (Section 9). Information theoretically, ℓ random coefficients are needed to blur the connection to an re-randomized level ℓ signature. We note this in the following theorem.

THEOREM 18.1. *Using* RE-RANDOM *on* $\sigma^{(\ell)}$ *we obtain uniform distribution of the signature elements where* $\sigma^{(\ell)} = (\sigma_0, \ldots, \sigma_\ell, \sigma_{-\ell}, \ldots, \sigma_{-1})$ *is any level* ℓ *signature on a message* m *valid for the public key* X_i.

PROOF. Consider two different level ℓ signatures $\sigma = (\sigma_0, \ldots, \sigma_\ell, \sigma_{-\ell}, \ldots, \sigma_{-1})$ and $\sigma' = (\sigma'_0, \ldots, \sigma'_\ell, \sigma'_{-\ell}, \ldots, \sigma'_{-1})$ on the same message m valid for the same public key X_i. To transform *any* σ_{-i} into σ'_{-i} for an $i \in \{1, \ldots, \ell\}$ exactly one coefficient is required. Therefore, in a transformation where each element σ_{-i} is transformed into σ'_{-i} for *all* $i \in \{1, \ldots, \ell\}$ exactly ℓ coefficients are needed. As we have seen in Part II, this also necessarily transforms the elements σ_i into σ'_i for $i \in \{0, \ldots, \ell\}$. This is exactly what RE-RANDOM $= \prod_{i=1}^{\ell}$ RE-RANDOM i does, it transforms a given level ℓ signature into another level ℓ signature which has the same distribution of random elements. □

Therefore we conclude that, information theoretically, the output of

$$\mathrm{ReSign}(\cdot, m, \ell - 1, \sigma^{(\ell-1)}, R_{ij}, X_i, X_j)$$

is indistinguishable from the output of

$$\mathrm{Sign}(\cdot, m, \ell, x_j)$$

where both algorithms output a level ℓ signature on m valid for the public key X_j. Using less coefficients cannot give us this uniform distribution.

We will first look at level ℓ signatures where one of the ℓ coefficients is 1 to analyze what happens when less coefficients are used. After that we will analyze the case where the random coefficients have a certain relation expressed by a linear equation. We will observe that in most of the cases the *unlinkability* property of the signature is lost.

Although the output of the algorithms $\mathrm{ReSign}(\cdot)$ from level $\ell - 1$ to level ℓ and $\mathrm{Sign}(\cdot)$ at level ℓ are indistinguishable we will consider these separately in each subsection. More precisely, in each subsection we consider the cases

- **Signing at level** ℓ, here the adversary tries to gather additional information from his knowledge.
- **Re-signing from level** $\ell - 1$ **to level** ℓ, here the adversary who has also the predecessor of the resulting level ℓ signature tries to link these two together.

Note that the discussion below considers only the *symmetric* pairing setting and therefore most of the results are not valid for the *asymmetric* setting (see Section 11). We address this problem at the end of this chapter.

18.1. Using lesser coefficients. We first look at what happens if one less random coefficient is used. Recall that $\boxed{\texttt{RE-RANDOM}} = \prod_{i=1}^{\ell} \boxed{\texttt{RE-RANDOM } i}$, in this section we assume that one $\boxed{\texttt{RE-RANDOM } i}$ was left out, ie. $r_i = 1$.

18.1.1. Signing at level ℓ**.** We first consider the output of the Sign$(\cdot)$ algorithm at level ℓ.

CASE $r_\ell = 1$**.** A signature signed at level ℓ with $r_\ell = 1$.

Consider Figure 18.1, the (red) encircled H is redundant since the element $\sigma_\ell^{(\ell)}$ is the same as the public key X_A. This means that an attacker can decouple this H, which gives him a level $\ell - 1$ signature on the same message valid for the same public key X_A. Thus, if the translation limit in the system was ℓ, this makes it possible to translate the signature once more than allowed.

CASE $r_1 = 1$**.** A signature signed at level ℓ with $r_1 = 1$. In Figure 18.2 we also see that the (red) encircled H is redundant, thus an attacker can again shorten this signature to a level $\ell - 1$ signature by removing $\sigma_{-1}^{(\ell)} = P$ and one of the $(r_\ell \cdots r_2)x_A P$ from the signature.

CASE $r_i = 1$**.** A signature signed at level ℓ with $r_i = 1$.

Here we also see in Figure 18.3 that the (red) encircled H is redundant so the elements $\sigma_{-i}^{(\ell)} = P$ and one of the $(r_\ell \cdots r_{i-1})x_A P$ can be removed.

Generally we observe that using lesser coefficients in the signing process is equivalent to signing the signature on a shorter level. As mentioned above this can be disadvantageous if there is a limitation on the number of allowed translations.

18.1.2. Re-signing from level $\ell-1$ **to level** ℓ**.** We now consider the output of the ReSign$(\cdot)$ algorithm when one less coefficient is used.

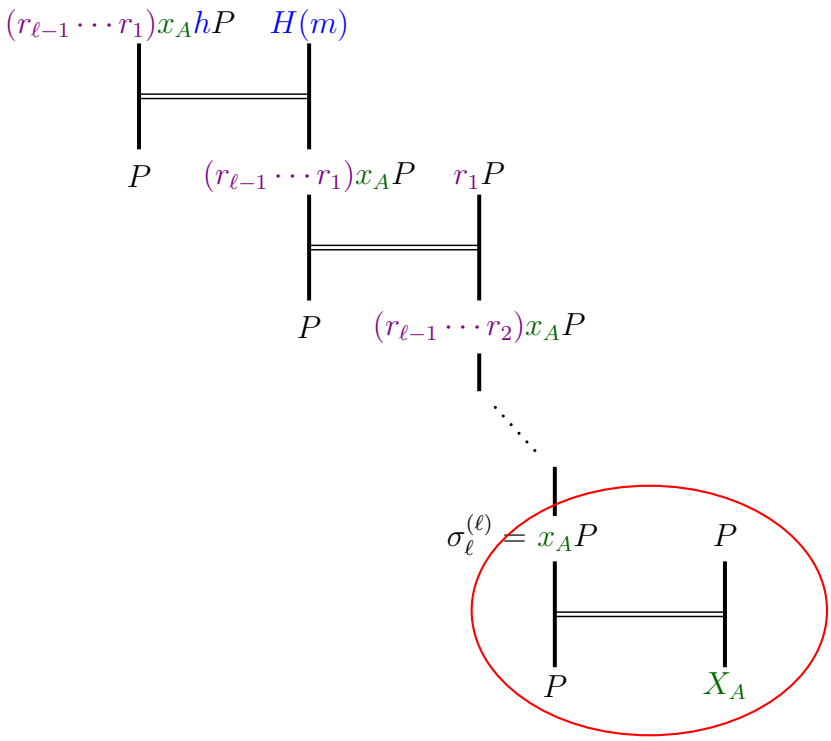


Figure 18.1: $\sigma^{(\ell)}$ with $r_\ell = 1$

CASE $r_\ell = 1$. We first consider a signature re-signed from level $\ell - 1$ to ℓ with $r_\ell = 1$. In this case as depicted in Figure 18.4 we observe that the last signer's public key X_A (on level $\ell - 1$) is visible in the signature since the new owner's public key (on level ℓ) is X_B. Taking away R_{AB} and X_B in the encircled H gives an attacker a level $\ell - 1$ signature $\sigma^{(\ell-1)}$ valid for the public key X_A, ie. the whole translation is lost. The attacker can also extract the re-signature key R_{AB}, thus also the *private proxy* property is lost. Since by assumption the attacker is also in possession of $\sigma^{(\ell-1)}$ the predecessor of $\sigma^{(\ell)}$, he can verify that $\sigma^{(\ell)}$ was most probably translated from $\sigma^{(\ell-1)}$ by checking

$$e\left(\sigma_{\ell-1}^{(\ell)}, \sigma_{-\ell+1}^{(\ell-1)}\right) \stackrel{?}{=} e\left(\sigma_{\ell-1}^{(\ell-1)}, \sigma_{-\ell+1}^{(\ell)}\right).$$

Thus, the *unlinkability* property of the signature is lost since the equivalence is given by

$$e\left(\sigma_{\ell-1}^{(\ell)}, \sigma_{-\ell+1}^{(\ell-1)}\right) = e\left(r_{\ell-1}\sigma_{\ell-1}^{(\ell-1)}, \sigma_{-\ell+1}^{(\ell-1)}\right)$$

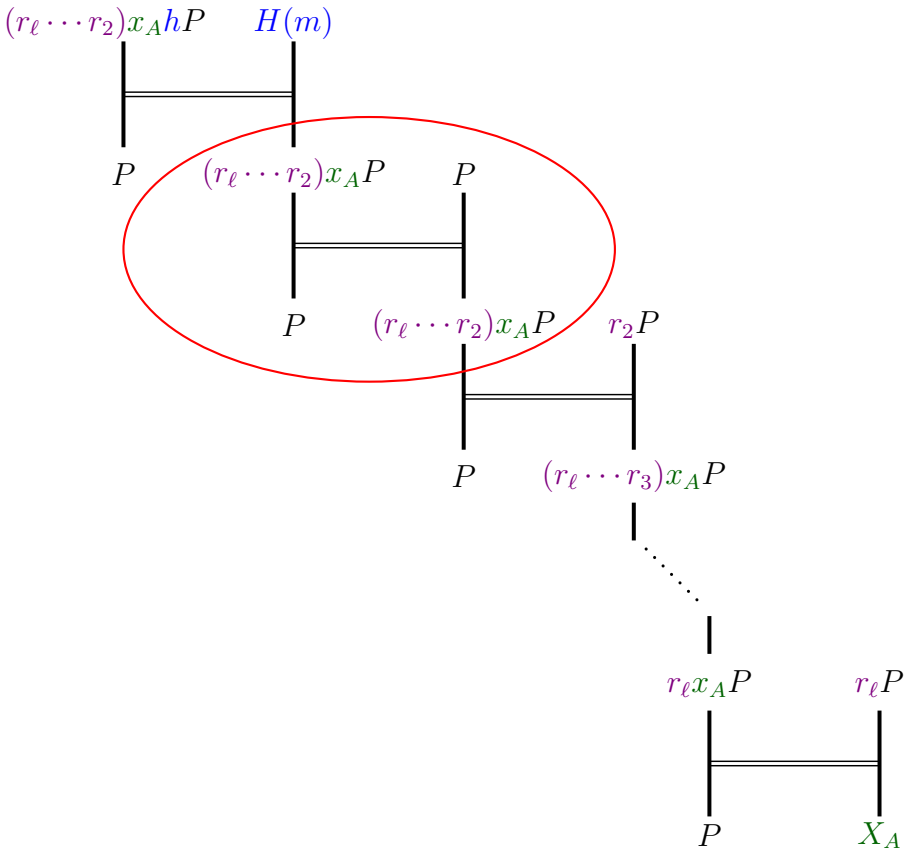


Figure 18.2: $\sigma^{(\ell)}$ with $r_1 = 1$

$$= e\left(\sigma_{\ell-1}^{(\ell-1)}, r_{\ell-1}\sigma_{-\ell+1}^{(\ell-1)}\right)$$
$$= e\left(\sigma_{\ell-1}^{(\ell-1)}, \sigma_{-\ell+1}^{(\ell)}\right).$$

CASE $r_1 = 1$**.** Here we consider a signature re-signed from level $\ell - 1$ to ℓ with $r_1 = 1$ as depicted in Figure 18.5. An attacker in possession of the predecessor $\sigma^{(\ell-1)}$ of $\sigma^{(\ell)}$ can see that $\sigma^{(\ell-1)}$ is most probably re-signed into $\sigma^{(\ell)}$ since

$$\sigma_{-1}^{(\ell)} = \sigma_{-1}^{(\ell-1)}.$$

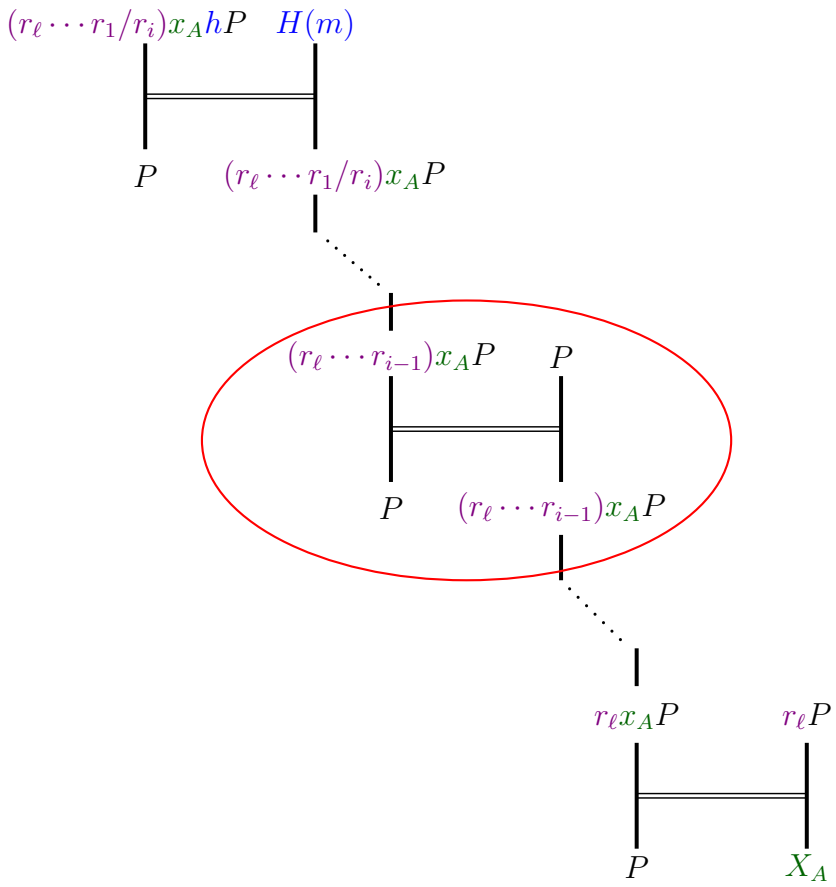


Figure 18.3: $\sigma^{(\ell)}$ with $r_i = 1$

He can verify this by checking

$$e\left(\sigma_1^{(\ell)}, \sigma_2^{(\ell-1)}\right) \stackrel{?}{=} e\left(\sigma_1^{(\ell-1)}, \sigma_2^{(\ell)}\right),$$

since the equivalence is given by

$$\begin{aligned} e\left(\sigma_1^{(\ell)}, \sigma_2^{(\ell-1)}\right) &= e\left((r_2 \cdots r_\ell)\sigma_1^{(\ell-1)}, \sigma_2^{(\ell-1)}\right) \\ &= e\left(\sigma_1^{(\ell)}, (r_2 \cdots r_\ell)\sigma_2^{(\ell-1)}\right) \\ &= e\left(\sigma_1^{(\ell-1)}, \sigma_2^{(\ell)}\right). \end{aligned}$$

Again the *unlinkability* property of the signature is lost.

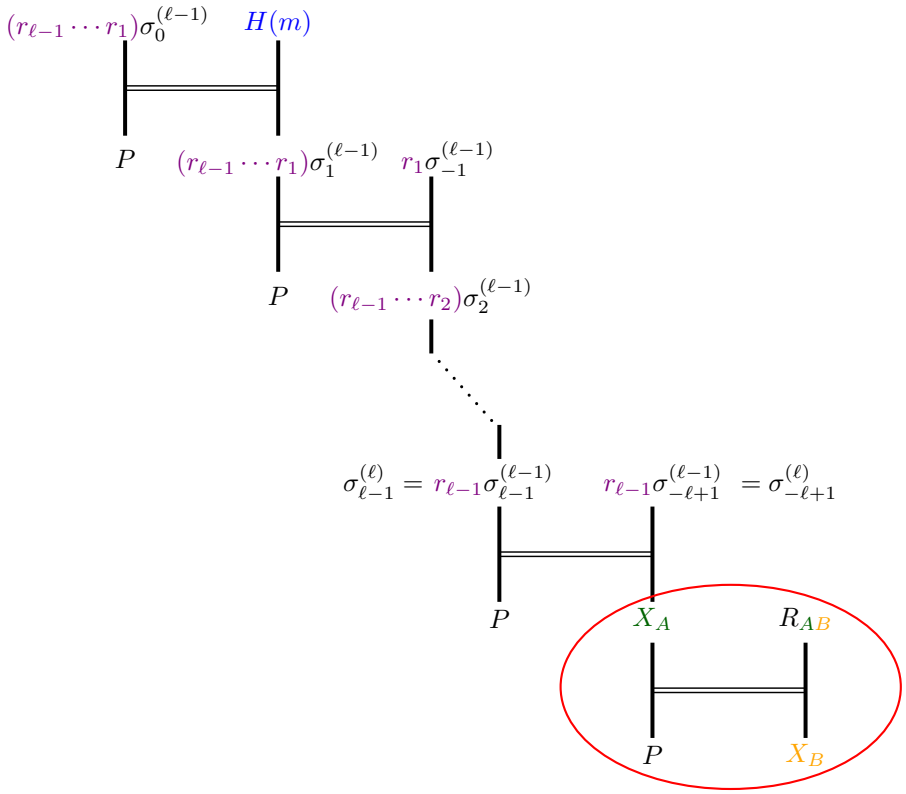


Figure 18.4: Re-Signing $\sigma^{(\ell-1)}$ to $\sigma^{(\ell)}$ with $r_\ell = 1$

CASE $r_i = 1$**.** A signature re-signed from level $\ell - 1$ to ℓ with $r_i = 1$. Generally if one of the coefficients $r_i = 1$ in the re-signing process then we have $\sigma^{(\ell)}$ as in Figure 18.6. An attacker in possession of the predecessor $\sigma^{(\ell-1)}$, can see that $\sigma^{(\ell-1)}$ was most probably re-signed into $\sigma^{(\ell)}$ since

$$\sigma_{-i}^{(\ell-1)} = \sigma_{-i}^{(\ell)}.$$

He can verify this by checking

$$e\left(\sigma_i^{(\ell)}, \sigma_{i+1}^{(\ell-1)}\right) \stackrel{?}{=} e\left(\sigma_i^{(\ell-1)}, \sigma_{i+1}^{(\ell)}\right),$$

because

$$e\left(\sigma_i^{(\ell)}, \sigma_{i+1}^{(\ell-1)}\right) = e\left((r_1 \cdots r_\ell / r_i)\sigma_i^{(\ell-1)}, \sigma_{i+1}^{(\ell-1)}\right)$$

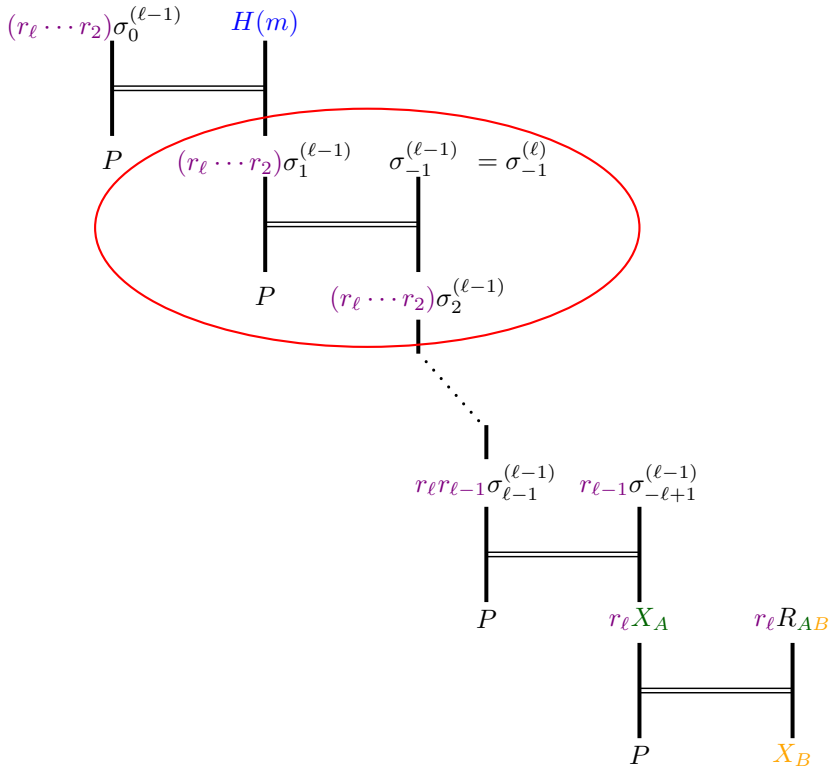


Figure 18.5: Re-Signing $\sigma^{(\ell-1)}$ to $\sigma^{(\ell)}$ with $r_1 = 1$

$$= e\left(\sigma_i^{(\ell-1)}, (r_1 \cdots r_\ell / r_i)\sigma_{i+1}^{(\ell-1)}\right)$$
$$= e\left(\sigma_i^{(\ell-1)}, \sigma_{i+1}^{(\ell)}\right).$$

Thus, using one coefficient less in the re-signing process always destroys the *unlinkability* property of the signature. Especially if $r_\ell = 1$ we loose the whole translation and also the *private proxy* property, since an attacker can easily access the re-signature key R_{AB}.

18.2. Related Coefficients. Now we analyze what happens if the coefficients satisfy a linear relation. As mentioned in the beginning, we assume that the relation is known to an attacker and we will try to find out if an attacker gains anything else from this knowledge. We start with the simple case where two coefficients are equal.

18.2.1. Signing at level ℓ.

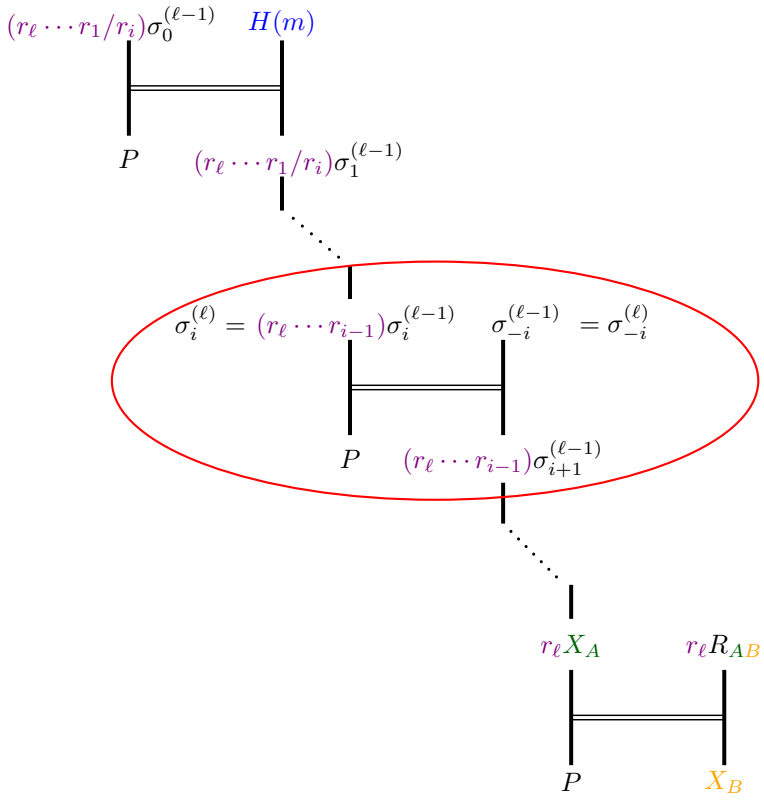


Figure 18.6: Re-Signing $\sigma^{(\ell-1)}$ to $\sigma^{(\ell)}$ with $r_i = 1$

CASE $r_\ell = r_{\ell-1}$**.** A signature signed at level ℓ with $r_\ell = r_{\ell-1}$.

In Figure 18.7 we see that $\sigma_{-\ell}^{(\ell)} = \sigma_{-\ell+1}^{(\ell)}$ which implies that $\sigma_\ell^{(\ell)} = r_{\ell-1}x_A P$ and $\sigma_\ell^{(\ell)} = r_{\ell-1}^2 x_A P$. However an attacker does not seem to gain much from this knowledge.

CASE $r_1 = r_2$**.** A signature signed at level ℓ with $r_1 = r_2$.

Similarly here in Figure 18.8 we have $\sigma_{-1}^{(\ell)} = \sigma_{-2}^{(\ell)} = r_2 P$ which also is not very useful for an attacker.

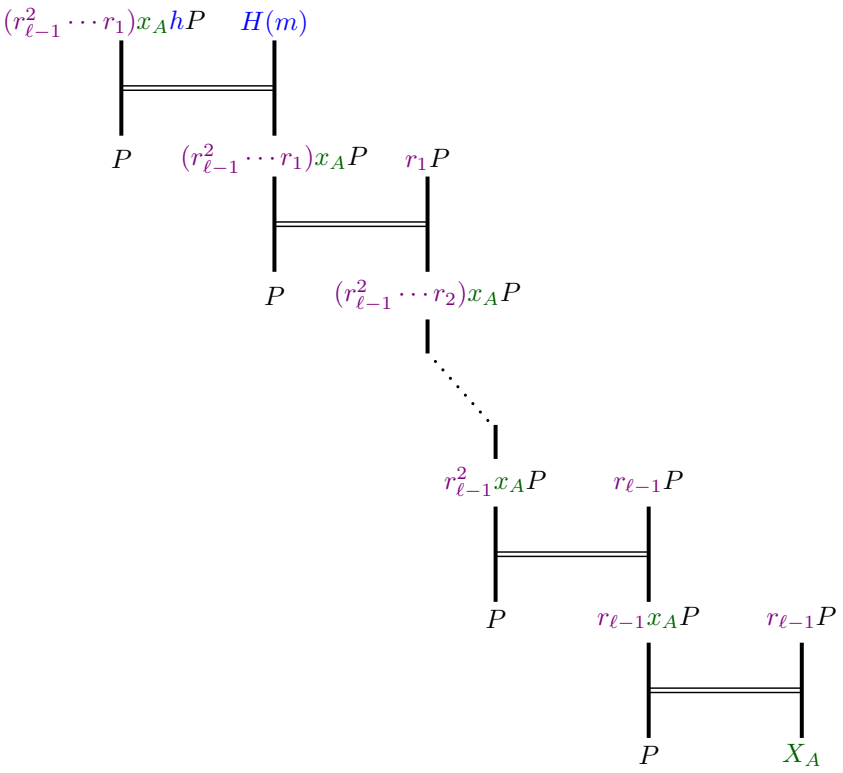


Figure 18.7: $\sigma^{(\ell)}$ signed with $r_\ell = r_{\ell-1}$

CASE $r_i = r_j$ for $i < j$. Generally we know that if a signature is signed on level ℓ with $r_i = r_j$, we would have two elements $\sigma_{-i}^{(\ell)} = \sigma_{-j}^{(\ell)}$ for $i, j \in \{1, \ldots, \ell\}$ (Figure 18.9).However it seems that this is not much of a use for an attacker.

GENERAL CASE $\prod_{i=1}^{\ell} r_i^{e_i} = 1$ for some exponents e_i. The general case where the relation of the coefficients is given as $\prod_{i=1}^{n} r_i^{e_i} = 1$ seems out of reach to be analyzed completely within the limits of this thesis. Nonetheless, we want to get an idea about this relation of the coefficients. Therefore we look into the next more complex case where $r_i = r_j \cdot r_k$ which is $r_i \cdot r_j^{-1} \cdot r_k^{-1} = 1$.

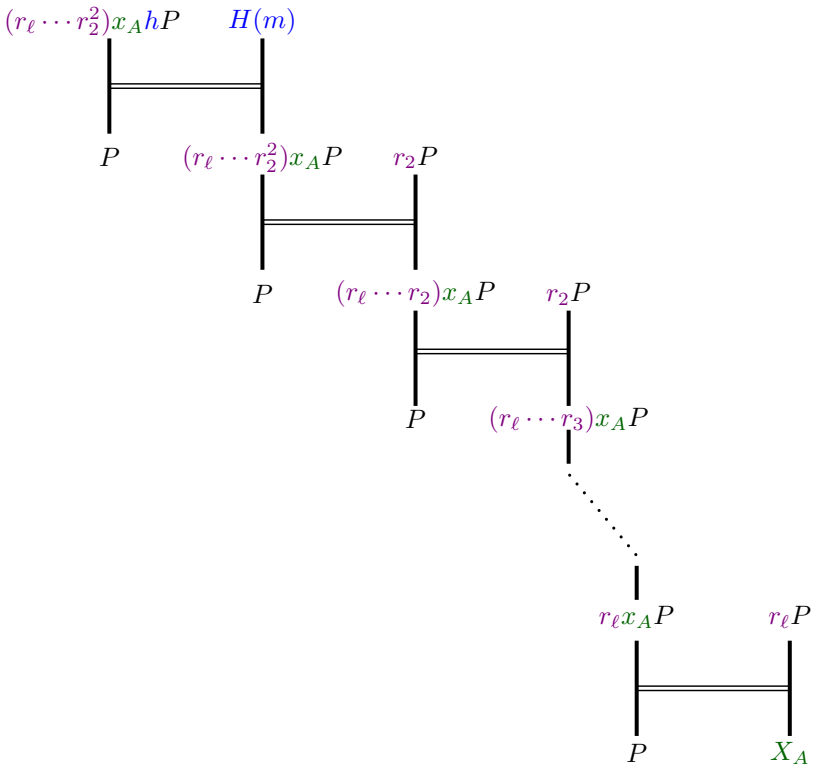


Figure 18.8: $\sigma^{(\ell)}$ signed with $r_1 = r_2$

The signature signed at level ℓ would contain the following elements:

$$\left[\sigma^{(\ell)}_{(-i)} = r_i P = r_j r_k P, \quad \sigma^{(\ell)}_{(-j)} = r_j P, \quad \sigma^{(\ell)}_{(-k)} = r_k P\right].$$

Assuming that the linear relation $r_i \cdot r_j^{-1} \cdot r_k^{-1} = 1$ is known to the attacker, he can verify this by checking

$$e(\sigma^{(\ell)}_{-i}, P) \stackrel{?}{=} e(\sigma^{(\ell)}_{-j}, \sigma^{(\ell)}_{-k}),$$

since

$$\begin{aligned} e(\sigma^{(\ell)}_{-i}, P) &= e(P, P)^{r_i}, \\ &= e(r_j P, r_k P), \\ &= e(\sigma^{(\ell)}_{-j}, \sigma^{(\ell)}_{-k}). \end{aligned}$$

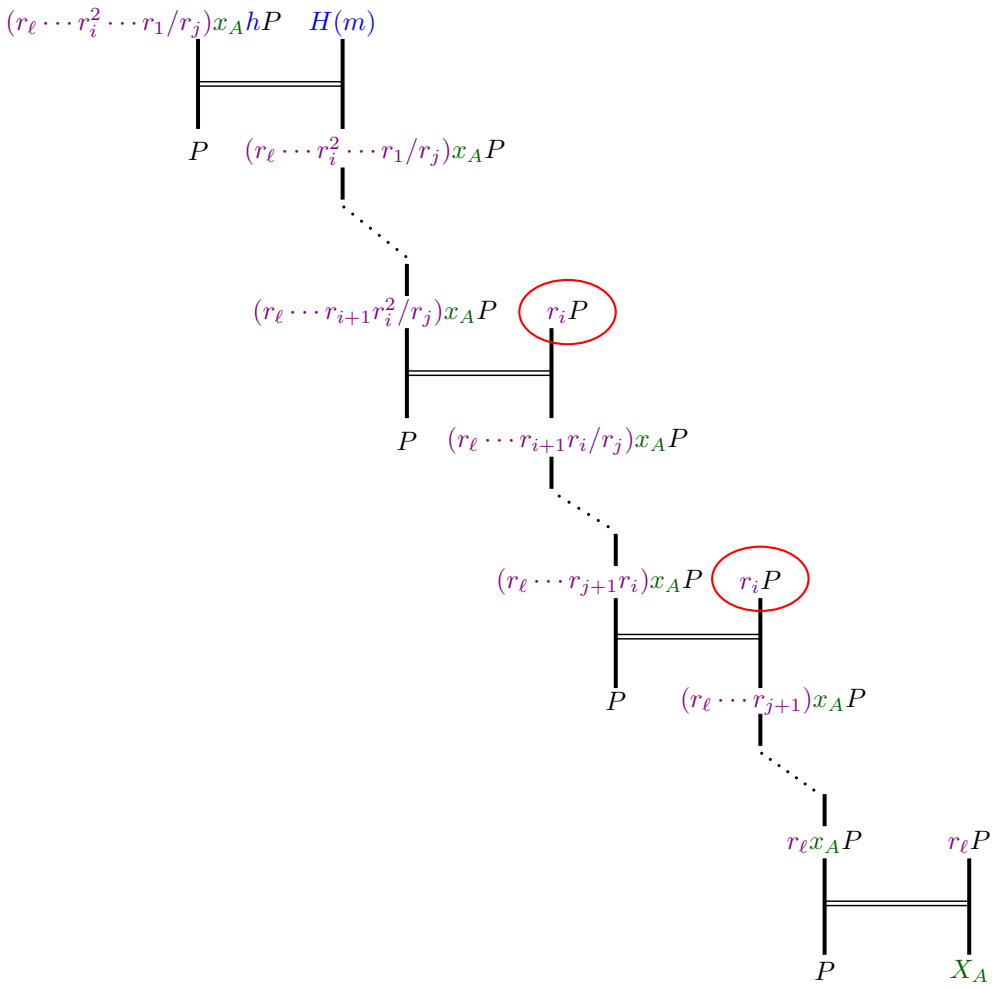


Figure 18.9: $\sigma^{(\ell)}$ signed with $r_i = r_j$

Here we see that we already reach some limit. The nature of the pairings $e(\cdot,\cdot)$, allows us to treat at most two $e_i = 1$ or one $e_i = 2$ and at most two $e_i = -1$ or one $e_i = -2$ at all. Anything else seems beyond the scope of group and pairing relations.

CASE $\sum_{i=1}^{n} \alpha_i \cdot r_i = 0$ **for** $n \leq \ell$**.** The other general case where the coefficients are additionally related to each other as $\alpha_1 r_1 + \alpha_2 r_2 + \cdots + \alpha_n r_n = 0$ can easily

be verified because

$$\sum_{i=1}^{n} \alpha_i r_i P = \sum_{i=1}^{n} \alpha_i \sigma_{-i} = 0 \cdot P = \mathbf{O}.$$

Note that by assumption the α_i's are known such that the values $\alpha_i r_i P = \alpha_i \sigma_{-i}$ can "easily" be calculated. Similarly by assumption the addition of these elements is also considered easy. We conclude that, if related coefficients are used in the signing process, an attacker is able to verify his knowledge of the relation but beyond that he does not seem to gain much from this knowledge.

18.2.2. Re-signing from level $\ell - 1$ to level ℓ.

CASE $r_\ell = r_{\ell-1}$**.** A signature re-signed from level $\ell - 1$ to ℓ with $r_\ell = r_{\ell-1}$. Considering Figure 18.10 we can easily link the signature to its predecessor by checking

$$e\left(\sigma_{\ell}^{(\ell)}, \sigma_{-\ell+1}^{(\ell-1)}\right) \stackrel{?}{=} e\left(X_A, \sigma_{-\ell+1}^{(\ell)}\right),$$

since

$$\begin{aligned} e\left(\sigma_{\ell}^{(\ell)}, \sigma_{-\ell+1}^{(\ell-1)}\right) &= e\left(r_{\ell-1} X_A, \sigma_{-\ell+1}^{(\ell-1)}\right) \\ &= e\left(X_A, r_{\ell-1} \sigma_{-\ell+1}^{(\ell-1)}\right) \\ &= e\left(X_A, \sigma_{-\ell+1}^{(\ell)}\right). \end{aligned}$$

This means that again we loose the *unlinkability* property of the signature.

CASE $r_1 = r_2$**.** A signature re-signed from level $\ell - 1$ to ℓ with $r_1 = r_2$. Consider Figure 18.11, we can link the signature to its predecessor by checking

$$e\left(\sigma_{-2}^{(\ell)}, \sigma_{-1}^{(\ell-1)}\right) \stackrel{?}{=} e\left(\sigma_{-2}^{(\ell-1)}, \sigma_{-1}^{(\ell)}\right),$$

since

$$\begin{aligned} e\left(\sigma_{-2}^{(\ell)}, \sigma_{-1}^{(\ell-1)}\right) &= e\left(r_2 \sigma_{-2}^{(\ell-1)}, \sigma_{-1}^{(\ell-1)}\right) \\ &= e\left(\sigma_{-2}^{(\ell-1)}, r_2 \sigma_{-1}^{(\ell-1)}\right) \\ &= e\left(\sigma_{-2}^{(\ell-1)}, \sigma_{-1}^{(\ell)}\right). \end{aligned}$$

Again the *unlinkability* property is lost.

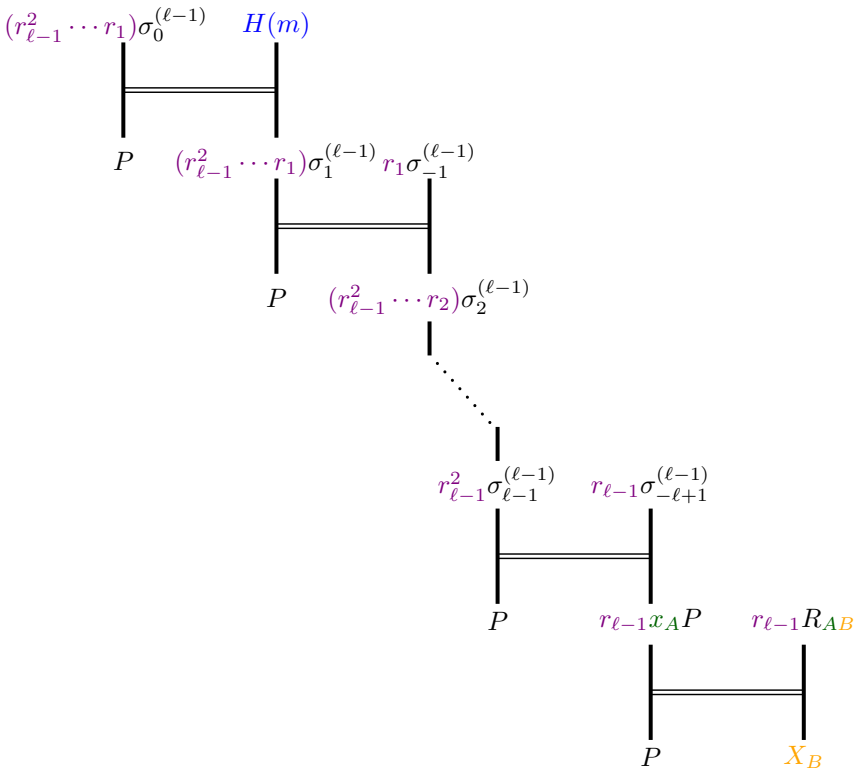


Figure 18.10: $\sigma^{(\ell-1)}$ re-signed to $\sigma^{(\ell)}$ with $r_\ell = r_{\ell-1}$

CASE $r_i = r_j$ for $i < j < \ell$. Consider Figure 18.12, as we have seen in the previous cases one could analogously link the level ℓ signature to its predecessor by checking

$$e\left(\sigma_{-i}^{(\ell)}, \sigma_{-j}^{(\ell-1)}\right) \stackrel{?}{=} e\left(\sigma_{-i}^{(\ell-1)}, \sigma_{-j}^{(\ell)}\right),$$

since

$$\begin{aligned} e\left(\sigma_{-i}^{(\ell)}, \sigma_{-j}^{(\ell-1)}\right) &= e\left(r_i\sigma_{-i}^{(\ell-1)}, \sigma_{-j}^{(\ell-1)}\right) \\ &= e\left(\sigma_{-i}^{(\ell-1)}, r_i\sigma_{-j}^{(\ell-1)}\right) \\ &= e\left(\sigma_{-i}^{(\ell-1)}, \sigma_{-j}^{(\ell)}\right). \end{aligned}$$

We conclude that, if two (or more) equal coefficients are used in the re-signing process the *unlinkability* property of the signature is lost.

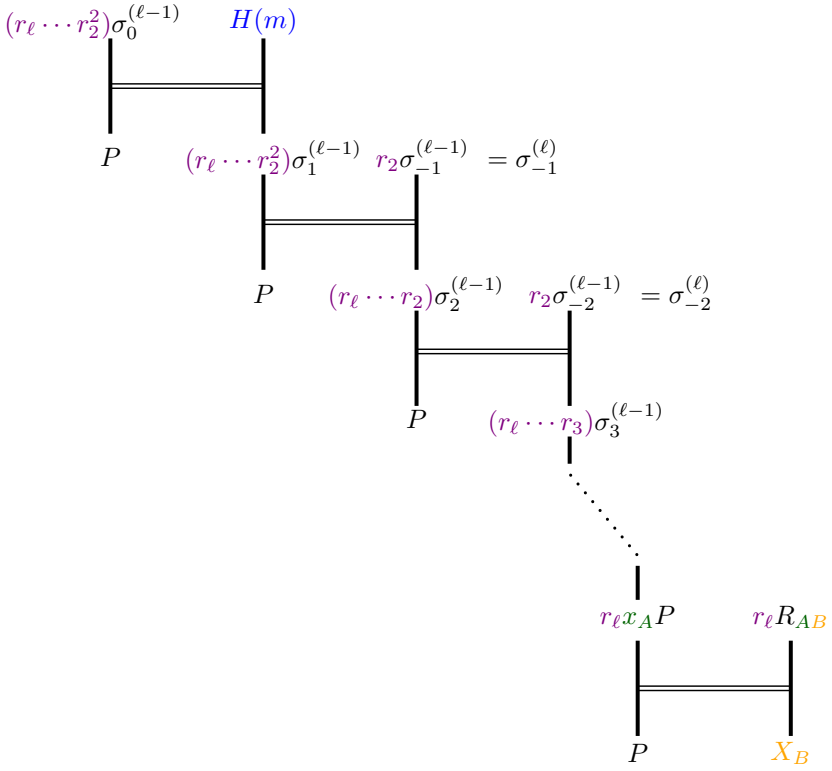


Figure 18.11: $\sigma^{(\ell-1)}$ re-signed to $\sigma^{(\ell)}$ with $r_1 = r_2$

GENERAL CASE $\prod_{i=1}^{n} r_i^{e_i} = 1$ for $n \leq \ell$. Again we only consider the next more complex case where $r_i = r_j \cdot r_k$. A signature which was re-signed from level $\ell - 1$ to level ℓ, contains the following elements:

$$\begin{aligned}
\sigma_{-i}^{(\ell)} &= r_i \sigma_{-i}^{(\ell-1)} = r_j r_k \sigma_{-i}^{(\ell-1)}, \\
\sigma_{-j}^{(\ell)} &= r_j \sigma_{-j}^{(\ell-1)}, \\
\sigma_{-k}^{(\ell)} &= r_k \sigma_{-k}^{(\ell-1)}.
\end{aligned}$$

Since we know that the elements $\sigma_{-i}^{(\ell-1)}, \sigma_{-j}^{(\ell-1)}$ and $\sigma_{-k}^{(\ell-1)}$ are different multiples of the base point P we can define them as three points P_i, P_j and P_k by setting

$$P_i := \sigma_{-i}^{(\ell-1)} = \gamma_i P,$$

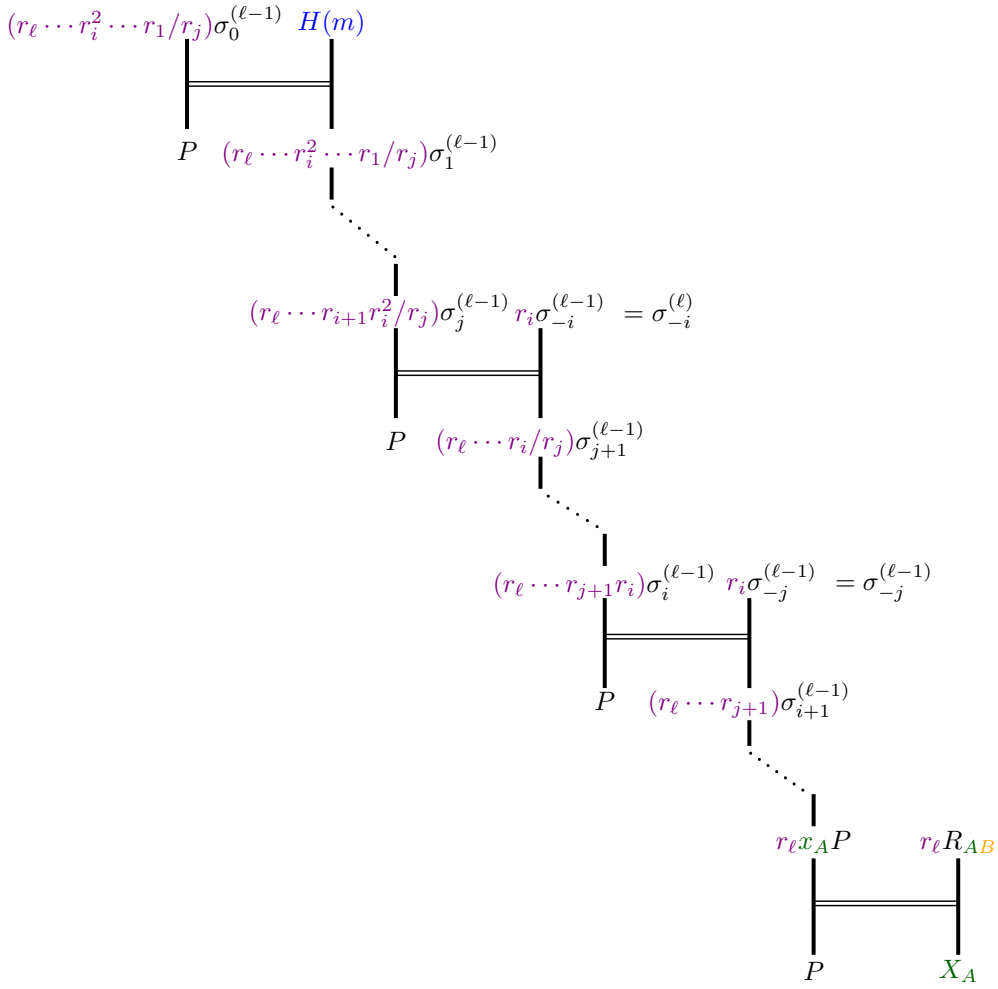


Figure 18.12: $\sigma^{(\ell)}$ signed with $r_i = r_j$

$$P_j := \sigma_{-j}^{(\ell-1)} = \gamma_j P,$$
$$P_k := \sigma_{-k}^{(\ell-1)} = \gamma_k P,$$

for some $\gamma_i, \gamma_j, \gamma_k \in \mathbb{Z}_p^\times$.

Now we can rewrite the re-signature elements $\sigma_{-i}^{(\ell)}, \sigma_{-j}^{(\ell)}$ and $\sigma_{-k}^{(\ell)}$ as different multiples of the base point P as

$$\sigma_i^{(\ell)} = r_i \sigma_{-i}^{(\ell-1)} = r_i P_i = r_i \gamma_i P = r_j r_k \gamma_i P,$$
$$\sigma_j^{(\ell)} = r_j \sigma_{-j}^{(\ell-1)} = r_j P_j = r_j \gamma_j P,$$

$$\sigma_k^{(\ell)} = r_k \sigma_{-k}^{(\ell-1)} = r_k P_k = r_k \gamma_k P.$$

For linking the signature to its predecessor we could try the following

$$\begin{aligned} e(r_i\gamma_i P, P) &= e(P,P)^{r_j r_k \gamma_i} \\ &= e(r_j r_k P, \gamma_i P) \\ &= e(r_j r_k P, \sigma_{-i}^{(\ell-1)}). \end{aligned}$$

or the other way around:

$$\begin{aligned} e(r_j\gamma_j P, r_k\gamma_k P) &= e(r_j r_k P, \gamma_j\gamma_k P) \\ &= e(r_i P, \gamma_j\gamma_k P), \end{aligned}$$

which is actually worse, since we do not know any of the values on the right hand side of the equations.

It seems that, to be able to link these two signatures, we at least have to calculate the value $r_j r_k P$ from given $r_j \sigma_{-j}^{(\ell-1)}$ and $r_k \sigma_{-k}^{(\ell-1)}$.

Another view point would be to fix the element $\sigma_{-i}^{(\ell-1)}$ as Q, and rewrite $\sigma_{-j}^{(\ell-1)}$ and $\sigma_{-k}^{(\ell-1)}$ relative to Q as

$$\begin{aligned} Q &= \sigma_{-i}^{(\ell-1)}, \\ aQ &= \sigma_{-j}^{(\ell-1)}, \\ bQ &= \sigma_{-k}^{(\ell-1)}. \end{aligned}$$

Recalling that

$$\begin{aligned} \sigma_{-i}^{(\ell)} &= r_i Q, \\ \sigma_{-j}^{(\ell)} &= r_j aQ, \\ \sigma_{-k}^{(\ell)} &= r_k bQ, \end{aligned}$$

our task is now to verify that $r_i = \frac{r_j a \cdot r_k b}{ab}$, with these elements. This would allow us to link the re-signature $\sigma^{(\ell)}$ to its predecessor $\sigma^{(\ell-1)}$. We define this as the 4-fold decisional Diffie-Hellman problem.

DEFINITION 18.2. ***4-fold decisional Diffie-Hellman problem*** *(4-DDH) is, given* $Q, aQ, bQ, \alpha Q, \beta Q, zQ \in \boldsymbol{G}$ *determine whether* $z = \frac{\alpha\beta}{ab}$.

Setting $r_i = z, r_j a = \alpha$ and $r_k b = \beta$ gives us the equivalence. To solve this 4-DDH instance in our pairing setting we can calculate the following values

$$\begin{aligned} e(aQ, bQ) &:= \xi, \\ e(aQ, r_k bQ) &:= \xi^{r_k}, \\ e(bQ, r_j aQ) &:= \xi^{r_j}, \\ e(abQ, r_i Q) &:= \xi^{r_j r_k}. \end{aligned}$$

This would reduce our task to solve the decisional Diffie-Hellman (DDH) instance in $\mathbf{G}_T$. This is, given ξ, ξ^m, ξ^n, ξ^t determine if $t = mn$. Note that to be able to do so we also need to calculate abQ from given Q, aQ, bQ which is a computational Diffie-Hellman (CDH) instance in $\mathbf{G}$. By assumption both of these instances are considered to be hard in the corresponding groups $\mathbf{G}$ and $\mathbf{G}_T$ respectively. Summarizing the results from above we get

$$\mathrm{CDH}(\mathbf{G}) + e(\cdot,\cdot) + \mathrm{DDH}(\mathbf{G}_T) \Rightarrow \text{4-DDH}(\mathbf{G}).$$

Thus, it is unlikely that an attacker knowing even the simplest form of the relation $r_i = r_j \cdot r_k$ is able to link the signature to its predecessor. Consequently it is also unlikely that an attacker with the knowledge of a more complex, general relation between coefficients is able to link a re-signature to its predecessor.

CASE $\sum_{i=1}^{n} \alpha_i \cdot r_i = 0$ for $n \leq \ell$. Beginning with a simple case assume that a relation as $r_i = r_j + r_k$ is known. Then, as we have seen before, a signature re-signed from level $\ell - 1$ to level ℓ would contain the elements

$$\begin{aligned} \sigma_{-i}^{(\ell)} &= r_i \sigma_{-i}^{(\ell-1)}, \\ \sigma_{-j}^{(\ell)} &= r_j \sigma_{-j}^{(\ell-1)}, \\ \sigma_{-k}^{(\ell)} &= r_k \sigma_{-k}^{(\ell-1)}. \end{aligned}$$

Again we know that the elements $\sigma_{-i}^{(\ell-1)}, \sigma_{-j}^{(\ell-1)}$ and $\sigma_{-k}^{(\ell-1)}$ are some different multiples of the base point P. As above we can rewrite the elements of the level ℓ signature as

$$\begin{aligned} \sigma_{-i}^{(\ell)} &= r_i \gamma_i P, \\ \sigma_{-j}^{(\ell)} &= r_j \gamma_j P, \\ \sigma_{-k}^{(\ell)} &= r_k \gamma_k P. \end{aligned}$$

Now if we try to link these elements we do the following

$$\begin{aligned} e(r_i\gamma_i P, P) &= e(P,P)^{r_i\gamma_i} \\ &= e\left((r_j + r_k)P, \gamma_i P\right) \\ &= e(r_j P, \sigma_{-i}^{(\ell-1)}) \cdot e(r_k P, \sigma_{-i}^{(\ell-1)}). \end{aligned}$$

As above we could try to calculate the values r_jP and r_kP from $r_j\sigma_{-j}^{(\ell-1)}$ and $r_k\sigma_{-k}^{(\ell-1)}$ respectively to link the signature to its predecessor.

Also, another view point is to fix $\sigma_{-i}^{(\ell-1)}$ as Q and rewrite $\sigma_{-j}^{(\ell-1)}$ and $\sigma_{-k}^{(\ell-1)}$ as aQ and bQ such that

$$\begin{aligned} Q &= \sigma_{-i}^{(\ell-1)}, & \sigma_{-i}^{(\ell)} &= r_i Q, \\ aQ &= \sigma_{-j}^{(\ell-1)}, & \sigma_{-j}^{(\ell)} &= r_j aQ, \\ bQ &= \sigma_{-i}^{(\ell-1)}, & \sigma_{-k}^{(\ell)} &= r_k bQ. \end{aligned}$$

Differing from above here our task would be to verify if $r_i = ab(r_j + r_k)$, for linking σ^{ℓ} to its predecessor $\sigma^{\ell-1}$. In our pairing setting we can calculate the values

$$\begin{aligned} e(aQ, bQ) &:= \xi, \\ e(aQ, r_k bQ) &:= \xi^{r_k}, \\ e(bQ, r_j aQ) &:= \xi^{r_j}, \\ e(abQ, r_i Q) &:= \xi^{r_j + r_k}. \end{aligned}$$

Given ξ, ξ^m, ξ^n, ξ^t we could easily determine if $t = m + n$ by calculating $\xi^m \cdot \xi^n$ and therefore link the re-signature $\sigma^{(\ell)}$ to its predecessor $\sigma^{(\ell-1)}$. However, as above we still need to calculate abQ from given Q, aQ, bQ which is again a CDH instance in $\mathbf{G}$.

Summarizing all the results from above, we see that the usage of so many coefficients is indeed necessary. Using less or even two equal coefficients, destroys the *unlinkability* property of the signature. Using related coefficients however does not seem to effect the security requirements of the signature. This is because an attacker knowing even the simplest form of a general relation of two coefficients, ie. $r_i = r_j \cdot r_k$ or $r_i = r_j + r_k$, is not able to link the signature to its predecessor or gain other useful knowledge against the security requirements. Unless of course he is able to solve at least the CDH problem. This means that, probably there is no harm in using related coefficients in the form of $\prod_{i=1}^{n} r_i^{e_i} = 1$ or in the form $\sum_{i=1}^{n} \alpha_i r_i$ for $n \leq \ell$, as long as two coefficients are not related to each other as $\alpha r_i = \beta r_j$. This implicates that a shorter signature is probably also sufficient.

18.3. The asymmetric setting. As noted in the beginning of this section most of the results achieved above are not valid for the *asymmetric* setting where the pairing has the form

$$e\colon \mathbf{G}_1 \times \mathbf{G}_2 \longrightarrow \mathbf{G}_T$$

for groups $\mathbf{G}_1, \mathbf{G}_2$ and $\mathbf{G}_T$. For example in the general case where two equal coefficients ($r_i = r_j$) are used in the re-signing process, we were able to link the re-signed signature $\sigma^{(\ell)}$ to its predecessor $\sigma^{(\ell-1)}$ by checking

$$e\left(\sigma_{-i}^{(\ell)}, \sigma_{-j}^{(\ell-1)}\right) \stackrel{?}{=} e\left(\sigma_{-i}^{(\ell-1)}, \sigma_{-j}^{(\ell)}\right).$$

Naturally, this does not work in the asymmetric setting because the elements in the pairing above would all be from the same group $\mathbf{G}_2$.

An idea to overcome this difficulty would be to look for an efficient mapping between $\mathbf{G}_1$ and $\mathbf{G}_2$ such that we could transform the elements of $\mathbf{G}_2$ into elements of $\mathbf{G}_1$ and vice versa when required. If we could find such a map, the results from above would also be valid for the asymmetric setting. However, finding such a map would also imply that the *decisional Diffie-Hellman* (DDH) assumption does not hold in the asymmetric setting. We recall the decisional Diffie-Hellman problem.

DEFINITION 18.3. *The* ***decisional Diffie-Hellman*** *problem is, given $P, aP, bP, cP \in \mathbf{G}$ to decide whether $c = ab$. The decisional Diffie-Hellman assumption is that this problem is hard to solve for certain groups $\mathbf{G}$.*

Obviously, in a pairing friendly group $\mathbf{G}$ with the symmetric setting this assumption does not hold since one can easily check

$$e(aP, bP) \stackrel{?}{=} e(cP, P).$$

As mentioned above, if an efficient map between $\mathbf{G}_1$ and $\mathbf{G}_2$ was found the DDH assumption would also not hold in the asymmetric setting. However, the DDH assumption is believed to be hard on ordinary pairing friendly elliptic curves in the asymmetric setting because no efficient maps seem to exists between $\mathbf{G}_1$ and $\mathbf{G}_2$ Freeman (2010). Therefore it seems that in the asymmetric setting the knowledge of the coefficient relations gives an attacker even less information than in the symmetric setting.

19. Conclusion

Concluding the results in this chapter, it seems that this type of construction of a *multi-use uni-directional proxy re-signature* does not allow much tweaking when it comes to efficiency. As already pointed out in the previous sections, shortening the signature seems hard to achieve although a shorter signature is probably sufficient for the same security requirements. Note that the information given to the proxy is strongly related to the *unlinkability*, *transparency* and the *private proxy* properties of the signature scheme and that this information is included into the new signature during the translation process. This inevitably will increase the size of the signature. This also increases the amount of randomness which is used to blind out the elements for achieving the unlinkability. This means that the amount of random coefficients is strongly related to the translation process of the signature.

Part V
Applications

In this section we will try to point out the possible applications of *proxy re-signatures* as motivated in the beginning of Part II. Although *proxy re-signatures* have been analyzed for their usages in e-cash systems by the "pairings and advances in cryptology for e-cash" PACE-Project (2008), the most promising application of *proxy re-signatures* was proposed for an interoperable *digital rights management*(DRM) architecture in Taban, Cárdenas & Gligor (2006). After discussing this proposal in detail, we will shortly consider some other usages of *proxy re-signatures*, as proposed in Ateniese & Hohenberger (2005) and Chow & Phan (2008).

20. Torwards an Interoperable Digital Rights Management System

The increasing availability of broadband internet connections, and the large variety of digital media such as music and video files, e-books and other digital content has made trading these items through DRM content providers a very lucrative business. The recent success stories like Apple's iTunes, mark the economic importance of online shopping for digital content. The popularity of smart phones, portable multimedia players, the next generation gaming consoles serving as media centers and the emerging market of home entertainment industry indicate that this business will grow even further in the next years. However, the lack of interoperability is a major factor for users to complain, since they cannot use the digital content on the device of their choice. In a survey carried out by INDICARE (2005), users polled that they were willing to pay a higher price for more usage rights and device interoperability. The survey concluded that "it certainly pays for digital music providers to offer flexible usage rights, sharing features, and to enable the usage of digital music on various devices". Consequently this lack of interoperability does not only concern end users but also digital content providers, since it slows down the growth of the industry and gives reasons for circumventing DRM mechanisms. Although there have been similar approaches to DRM interoperability as in Koenen *et al.* (2004) and Kravitz & Messerges (2005), the most satisfying approach is the one of Taban *et al.* (2006), which we analyze now.

20.1. Proposed Architecture. The architecture in the proposal consists of the commonly accepted model for the home networks consisting of

- **Content providers (CP)** who provide digital content to consumers protected by their own DRM mechanisms.
- **Consumer electronics (CE) operators** who provide electronic devices to users and guarantee the DRM capability and the soundness of their devices against manipulations, for example with a trusted platform module (T.C.G. 2008).
- **Licensing organizations** who certify and manage compliant devices. These organizations also manage revocation lists of compromised or circumvented devices.
- **Domain interoperability manager (DIM) operators** who manufacture and sell devices that allow interoperability between different content providers.
- **Home network** consists of one single domain where consumers want to use the digital content they have purchased on different devices.

Consider Figure 20.1, the interoperability problem deals with two different devices D_A and D_B, for content providers P_A and P_B respectively. The domain interoperability manager (DIM) transfers the digital content available for D_A by provider P_A into the one of P_B used by D_B.

20.2. Interoperability Framework. For the entities defined above we now explain their roles and the trust relationships between them. This model also serves as a guarantee for all participants when a new party joins the system.

- **Licensing Organizations** act like a certificate authorities with well known public keys. The licensing organizations certify the CE and the DIM operators and keep lists of compromised and circumvented devices.
- **CE operators** certified by the licensing organization are bound to manufacture and sell only compliant devices. These devices store the certificates for the CE issued by the licensing organization. Such a certificate authorizes the public key PK_{D_A} of the device D_A as well as the public key PK_A of the content provider A.

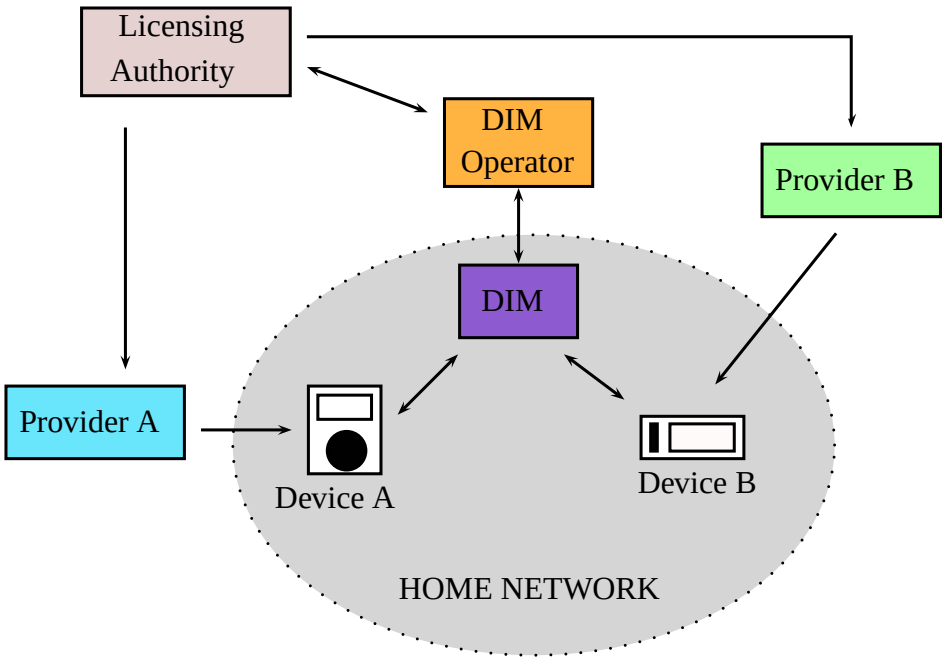


Figure 20.1: Interoperable DRM System

- **Content Providers** deliver digital content to authenticated devices after ensuring that the devices receiving digital content are not compromised or revoked by the licensing organizations. At the end of a delivery through secure communication channels a device D_A stores the encrypted content $\left(\{M\}_{CEK}, \quad L = \{CEK, R\}_{PK_{D_A}}, \quad \sigma_{PK_A}\right)$, where L is the license containing the content encryption key CEK and the rights R associated with the content M encrypted with PK_{D_A} the public key of device D_A, and as well as the signature σ_{PK_A} signed by DRM content provider A on the license L verifiable with the public key PK_A of the content provider A.

- **Domain Interoperability Manager (DIM)** is the heart piece of the interoperability framework. It stores the re-encryption key RE_{AB} and the re-signature key RS_{AB} for translation between DRM providers A and B. It also stores translation policies on which the DRM operators have

agreed before. These policies can contain how many times a file can be translated, how many devices should be accepted, etc. The assumption is that the DIM has at least periodical internet connectivity which provides it with the necessary tools and updates for translating between various DRM providers and their devices. Besides these functions the DIM can also provide the user with the necessary information of his rights on different content and devices. It can inform the users of their options regarding the purchased digital content.

20.3. Cryptographic Tools. The two main tools used to achieve this proposed framework are *proxy re-encryption* (Part I) and *proxy re-signatures*. As mentioned before, a *proxy re-encryption* scheme allows a semi trusted proxy to translate a ciphertext C_{PK_A} computed under the public key PK_A of `Aylin` into a ciphertext C_{PK_B} that can only be decrypted with the secret key SK_B of `Boris`. On the other hand a *proxy re-signature* scheme allows a semi trusted proxy to translate a signature σ_A valid for the public key PK_A of `Aylin` into a signature σ_B on the same message valid for the public key PK_B of `Boris`.

As mentioned above a device D_A of the content provider A stores a digital content $\{M\}_{CEK}$, a license $L = \{CEK, R\}_{PK_{D_A}}$ encrypted with the public key PK_{D_A} of device D_A, a signature σ_{PK_A} on $\{CEK, R\}$ signed by the content provider A and as well as the public key PK_A of the content provider A. To access the content the device D_A first verifies the license L with the public key PK_A, then decrypts the license L with its secret key SK_{D_A} to obtain the content encryption key CEK which is used to access the digital content M. Usually its safe to assume that CEK is used with a symmetric encryption algorithm because of it is efficiency. Note that we also assume that the device cannot be compromised or at least it will be revoked when it is compromised.

20.4. Proposed Protocols. In this framework the DIM acts as a semi trusted proxy which by assumption already has the re-encryption and re-signature keys RE_{AB} and RS_{AB}, respectively. As mentioned before this can be achieved with (at least) periodical internet connectivity. Based on the trust and DRM realizations of the content providers, two different protocols are proposed.

20.4.1. Protocol 1. This protocol minimizes the providers' trust in the DIM by disallowing him the access to the unencrypted content M. The disadvantage of this is that the exporting device D_A and the importing device D_B must render similar DRM formats. This means that the different DRM systems must use similar encryption and signature algorithms as well as similar rights expression

languages. Nonetheless this protocol can be used for for different devices of the same content provider or for *multi* devices which support more than one providers DRM mechanisms.

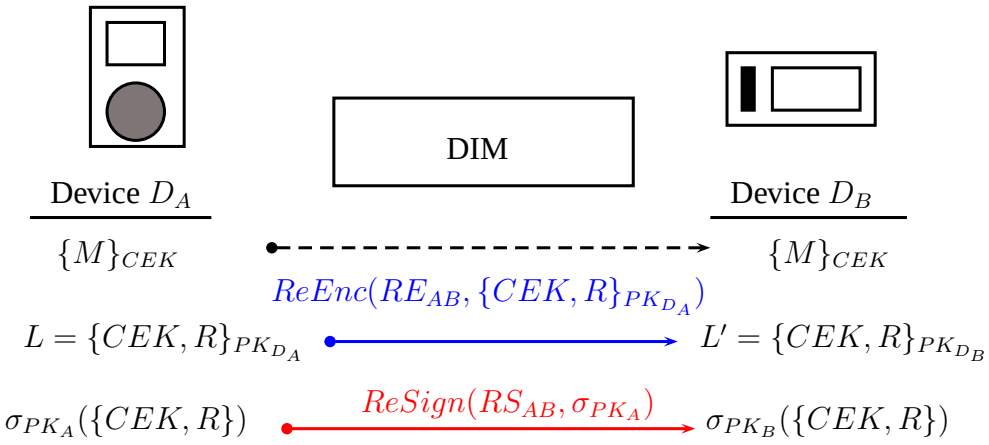


Figure 20.2: Protocol 1

Consider Figure 20.2, the DIM will first re-encrypt the license $L = \{CEK, R\}_{PK_{D_A}}$ which can be decrypted using the secret key SK_{D_A} of device D_A to $L' = \{CEK, R\}_{PK_{D_B}}$ which later can be decrypted with the secret key SK_{D_B} of device D_B. The DIM then will re-sign the signature σ_{PK_A} to σ_{PK_B} on the same *unencrypted* license $\{CEK, R\}$ which is now verifiable with the public key PK_B of the content provider B. Note that the DIM cannot verify the signature of the unencrypted license since it has only access to the encrypted content $\{M\}_{CEK}$ and the encrypted license $\{CEK, R\}_{PK_{D_A}}$.

20.4.2. Protocol 2. This protocol allows interoperability of content providers with different DRM mechanisms. Specifically this protocol supports diversity in content formats, right expression languages and encryption algorithms. The only assumption is that the exporting and importing devices support the same signature scheme. In this protocol the exporting device D_A and the DIM agree on a session key k, to avoid the disclosure of the secret key SK_{D_A} of device D_A. Device D_A then decrypts L and encrypts it again with the session key k. Then the DIM does the following:

1. Receive $\left(\{M_A\}_{CEK_A},\ L = \{CEK_A, R_A\}_k,\ \sigma_{PK_A}\right)$ and δ_{PK_A}, where M_A

is the digital content encoded with the format of the content provider A, CEK_A is the content encryption key provided by A, R_A the rights expression language of A, σ_{PK_A} the signature of A on (CEK_A, R_A) and δ_{PK_A} is a signature of A on (ID_{M_A}, ID_{R_A}) the identifiers of the digital content and the associated rights.

2. The DIM first decrypts $L = \{CEK_A, R_A\}_k$ to obtain CEK_A and R_A, and obtains M_A by decrypting $\{M_A\}_{CEK_A}$ with CEK_A.

3. Then the DIM transcodes M_A into the format M_B of the content provider B and translates R_A into the rights expression language R_B of provider B.

4. After generating a new content encryption key CEK_B the DIM encrypts M_B with the symmetric encryption algorithm of provider B and obtains $\{M_B\}_{CEK_B}$. Now the DIM calculates a new license $L' = \{CEK_B, R_B\}_{PK_{D_B}}$ by encrypting it with the public key PK_{D_B} of device D_B.

5. The DIM now signs $\{CEK_B, R_B\}$ by using its own secret key SK_{DIM} with a certificate *cert* issued by the content provider B that certifies the public key PK_{DIM} and obtains σ_{PK_B}.

6. Finally the DIM re-signs the signature δ_{PK_A} using the the re-signature key R_{AB} and gets δ_{PK_B}. This is signature is used to assure that the transferred content is authentic and original.

Now the importing device B receives $\{M_B\}_{CEK_B}$, $L = \{CEK_B, R_B\}_{PK_{D_B}}$, σ_{PK_B}, δ_{PK_B} which is a valid content for the DRM provider B with the correct encoding of the file M and the rights expression language R_B of B. In this protocol the transcoding and the translation process can be very time consuming especially for lager files. Also, as mentioned in the beginning, this process requires a greater amount of trust in the DIM from content providers because the DIM has access to the unencrypted content M.

20.5. Security. Traditionally there are three types of attacks for DRM systems. These are attacks against:

1. the DRM protocols,

2. the client devices, ie. their secure storage,

3. and the rendering application.

The purpose of these attacks is to obtain the unencrypted digital content. In an attack against the DRM protocol, the attacker tries to exploit a weakness in the design or implementation of the protocol. In an attack against the client devices, the attacker tries to corrupt the device such that he can get access to the raw content. In an attack against the rendering application, the attacker tries to obtain the unencrypted content while rendering it. Beyond this in an interoperable DRM architecture the authors define three new attacks. These are:

1. the cross-compliancy of devices,
2. splicing of content with an illegitimate license,
3. and leakage of content or content encryption keys on the migration path.

The most important concern for interoperability is that the attackers will discover vulnerabilities of certain implementations for compromising devices. To obviate this, providers need to assure that all devices are up to date and compliant. The reasonable assumption is that the home network has at least periodical internet connectivity such that content providers can check if a device is up to date before delivering digital content. This is complicated since the DIM and the importing devices also need to be up to date and the content transfer between devices can also happen off line.

In the second threat scenario the concern is that an attacker can obtain a license from a possibly corrupted device and modifies the license accordingly or produces another one to get access to the raw content.

The third threat scenario is that an eavesdropper can learn secret information from the interoperability protocol itself. Furthermore a corrupted device can also be used to extract secret information from compliant devices.

20.5.1. Security of protocol 1. As mentioned above this protocol allows minimal trust to the DIM. The DIM is used as a semi trusted proxy who translates the signatures and the ciphertext. Thus, if an attacker compromises the DIM the best he can do is to extract the re-signature and re-encryption keys which don't give him any further advantage. Since the DIM does not have access to the unprotected content, the attacker does not gain anything during the translation process. Therefore the only way to attack the system is to break the underlying cryptographic assumptions.

20.5.2. Security of protocol 2. Any attacker corrupting the DIM cannot gain any information on the system secrets because the DIM does neither store

the secret keys of content providers nor of devices. Furthermore to prevent attackers from simply generating their own re-signature and re-encryption keys, the DIM is validated through an initial registration with the various content providers.

The biggest concern in protocol 2 is the fact that the DIM has access to unprotected content, either directly when decrypting the content or indirectly through the decryption of the license. Therefore the content providers have to make sure that the DIM is working on trusted computing platform (T.C.G. 2008) to ensure that the translated content is not leaked during the whole translation process.

For further considerations about security, ideas about attestation of compliant devices, attack scenarios and a more detailed insight of this proposal we refer to the original publication Taban *et al.* (2006).

21. Masking the Internal Structure of a Company

Recall one of the motivations in the beginning of Part II, a company with different working groups each mandated by its own supervisor. When a (sub)project is finished the supervisor signs it with his own private key. Then the signature of the supervisor is translated (via a proxy) into the signature of the company. For example, an automobile company does not manufacture all the parts of their cars, more likely the company has its own contractors (other companies) which are specialized in manufacturing specific parts. The manufactured parts are sent to the company's different divisions like fabrication, service, spare parts sales, etc. The parts in possession of these different divisions which are possibly distributed all over the world must all have a valid signature of the car company itself. Here we can safely assume that modern day logistics in this scale uses RFID chips to deploy signatures. However these chips usually cannot be re-programmed and have limited storage space. This rules out the trivial solutions mentioned in Part II. Besides that, the car company could also delegate its signing rights to its different branches, however this would also increase the chance of misuse. In addition to this, due competitive reasons the car company is interested in keeping its supply partners confidential. Thus, a *proxy re-signature* scheme can be very useful in this kind of global and distributed setting. The different divisions, after receiving and verifying the quality of the manufactured parts, translate the signatures of the manufacturers into valid signatures of the car company. The different divisions act as a proxy between the part manufacturers and the car company.

22. E-Passport Systems and Graphs

Proxy re-signature schemes can also be used in immigration and customs services for travelers with machine readable travel documents like e-passports. The signature within the e-passport can be transformed while the traveler passes through different checkpoints. This process ensures that the traveler goes through all the required checkpoints while only one signature is kept within the passport. In general, *proxy re-signatures* can be used to ensure that a certain path in a graph is taken. This can be achieved by simply providing each node in the graph, except the first one, with the re-signature keys but no signing keys, such that each node is only able to translate signatures of adjacent nodes. Consider Figure 22.1, the signer A generates the signature $\sigma_A(m)$ for the message m, along the path the intermediate nodes act as proxies and transform the signature into its final version $\sigma_D(m)$ which is then verified by V.

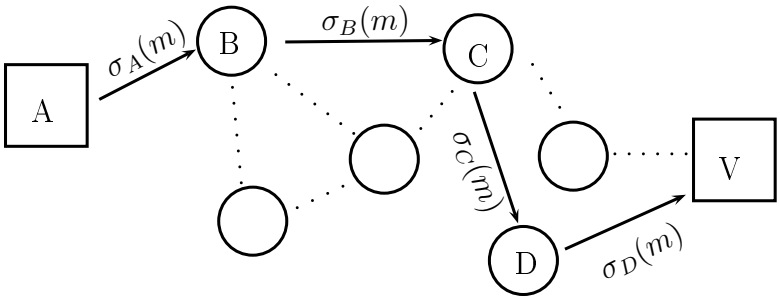


Figure 22.1: An authenticated path

We note that even if one or more nodes are compromised they still cannot produce signatures of their own, thus message injection is not possible at any time. Since only one signature needs to travel the path, there is no need for accumulating signatures and public keys along the path. Recall the *unlinkability* property of the signature scheme, a full path can be kept secret since each node on the path has only the information of the preceding node. This setting can be very useful in networks where the trustworthiness of all nodes is not given.

23. Certificate Management

The certification of public keys is usually implemented as a signature from the certificate authority (CA) on the public key belonging to a specific identity.

Naturally the certification of new public keys is a time consuming and expensive process. These certificates are often deployed in networks to allow transactions between users. Now consider two different networks where the users only trust their certification authority due to security reasons. When two users `Aylin` and `Boris` in different networks want to communicate with each other, they will first exchange their certified public keys. However, by assumption `Aylin` only trusts CA_1 and `Boris` only CA_2 such that they cannot verify the identities of each other. In this setting *proxy re-signatures* could be very useful since the different certification authorities could set up proxies to allow the translation of their signatures. Generalizing this, *proxy re-signatures* can be used for transparent cross-certification between different certificate authorities such that certificates of one authority can be converted into certificates of others.

Part VI
Conclusion

The research field of digital signatures is diverse and very fascinating. Different requirements and application areas, led to various types of digital signatures some of which we discussed at the end of Part I. Specially when it comes to practical applications in digital rights and e-cash systems, *proxy re-signatures* can be very useful because of their translation property. In this context we introduced and analyzed the *multi-use unidirectional proxy re-signature* proposed by Libert & Vergnaud, in detail. We first introduced the scheme step by step to provide a comprehensive understanding of the signing and translation process before writing down the formal notations.

In Part III, we reviewed the cryptographic assumptions, the adversary model and the two different simulation environments. We also introduced a new security definition which overcomes the outlined shortcomings of original security definition from Ateniese & Hohenberger (2005). We observed that our new security definition does not only include all the requirements listed in Libert & Vergnaud (2008a) but also provides the necessary flexibility to be adapted and used for different requirements. This brought us to the conclusion that our new security definition can be used in the future to prove the security of different proxy re-signature schemes. We finished this chapter with a detailed proof of security in the random oracle model and in the standard model after a slight modification of the signature scheme.

In Part IV, we have seen that the amount of randomness used in each translation step, is necessary to preserve the *unlinkability* property of the signature. We also introduced a new problem class, the *chain shortening* problem, which we used to analyze the length of the signature. We observed that if shortening the signature was somehow possible this would almost mean solving the CDH. Thus, we concluded that if the proxy has to insert some information (even a single bit) into the signature, we would inevitably end up with signature which grows with the number of delegations.

In Part V we pointed out some usages of *proxy re-signatures*, specially the proposed interoperable DRM architecture from Taban *et al.* (2006). Despite these practical applications of *proxy re-signatures*, discussed in that chapter, it is still desirable to have a *multi-use proxy re-signature* scheme which combines the identities of the content provider and the user purchasing it. In such a scheme, when `Aylin` purchases a file m from the content provider `Peter`, a combined signature σ_{PA} of `P` and `Aylin` should authenticate the document m.

If now `Aylin` wants to give away the file to `Boris`, the translation property should allow the translation of σ_{PA} into σ_{PB} a combined signature of `Boris` and `Peter`. Ideally of course this should happen without the interaction of the content provider `Peter`. This is because, as in a flea market example, only `Aylin` and `Boris` have interact for trading. In the research area of content protection and e-cash systems constructing such a signature could be very innovative and useful.

References

Jee Hea An, Yevgeniy Dodis & Tal Rabin (2002). On the Security of Joint Signature and Encryption. In *EUROCRYPT 2002*, Lars R. Knudsen, editor, volume 2332 of *Lecture Notes in Computer Science*, 83–107. Springer. ISBN 3-540-43553-0.

G. Ateniese, K. Fu, M. Green & S. Hohenberger (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)* **9**(1), 1–30. ISSN 1094-9224.

Giuseppe Ateniese & Susan Hohenberger (2005). Proxy re-signatures: new definitions, algorithms, and applications. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, Vijay Atluri, Catherine Meadows & Ari Juels, editors, 310–319. ACM. ISBN 1-59593-226-7.

Elaine Barker, William Barker, William Burr, William Polk & Miles Smid (2007). Recommendation for key management - part 1: General (revised). In *NIST Special Publication 800-57*.

Elaine Barker, William Burr, Alicia Jones, Timothy Polk, Scott Rose, Quynh Dang & Miles Smid (2009). Recommendation for key management - part 3: Application-Specific Key Management Guidance. In *NIST Special Publication 800-57*.

Paulo S. L. M. Barreto (2009). The Pairing-Based Crypto Lounge. URL `http://www.larc.usp.br/~pbarreto/pblounge.html`. [Online; accessed 13-April-2011].

Paulo S. L. M. Barreto & Michael Naehrig (2005). Pairing-Friendly Elliptic Curves of Prime Order. In *Selected Areas in Cryptography*, Bart Preneel & Stafford E. Tavares, editors, volume 3897 of *Lecture Notes in Computer Science*, 319–331. Springer. ISBN 3-540-33108-5.

Mihir Bellare & Thomas Ristenpart (2009). Simulation without the Artificial Abort: Simplified Proof and Improved Concrete Security for Waters' IBE Scheme. Cryptology ePrint Archive, Report 2009/084. `http://eprint.iacr.org/`.

Mihir Bellare & Phillip Rogaway (1993). Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, 62–73.

Matt Blaze, Gerrit Bleumer & Martin Strauss (1998). Divertible Protocols and Atomic Proxy Cryptography. In *Advances in Cryptology EUROCRYPT - 98*, Kaisa Nyberg, editor, volume 1403 of *Lecture Notes in Computer Science*, 127–144. Springer Berlin / Heidelberg.

Dan Boneh, Ben Lynn & Hovav Shacham (2004). Short Signatures from the Weil Pairing. *Journal of Cryptology* **17**(4), 297–319. ISSN 0933-2790.

Ran Canetti, Oded Goldreich & Shai Halevi (2004). The random oracle methodology, revisited. *J. ACM* **51**(4), 557–594. ISSN 0004-5411.

Ran Canetti & Susan Hohenberger (2007). Chosen-ciphertext secure proxy re-encryption. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, 185–194. ACM, New York, NY, USA. ISBN 978-1-59593-703-2.

D. Chaum (1996). Private signature and proof systems. *US Patent 5,493,614* .

Sherman Chow & Raphael Phan (2008). Proxy Re-signatures in the Standard Model. In *Information Security*, Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen & Der-Tsai Lee, editors, volume 5222 of *Lecture Notes in Computer Science*, 260–276. Springer Berlin / Heidelberg. URL `http://dx.doi.org/10.1007/978-3-540-85886-7_18`. 10.1007/978-3-540-85886-7_18.

Sherman Chow, Jian Weng, Yanjiang Yang & Robert Deng (2010). Efficient Unidirectional Proxy Re-Encryption. In *Progress in Cryptology – AFRICACRYPT 2010*, Daniel Bernstein & Tanja Lange, editors, volume 6055 of *Lecture Notes in Computer Science*, 316–332. Springer Berlin / Heidelberg. URL `http://dx.doi.org/10.1007/978-3-642-12678-9_19`. 10.1007/978-3-642-12678-9_19.

Jean-Sébastien Coron (2000). On the Exact Security of Full Domain Hash. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '00, 229–235. Springer-Verlag, London, UK. ISBN 3-540-67907-3. URL `http://portal.acm.org/citation.cfm?id=646765.704121`.

Craig Costello & Douglas Stebila (2010). Fixed Argument Pairings. In *LATINCRYPT*, Michel Abdalla & Paulo S. L. M. Barreto, editors, volume 6212 of *Lecture Notes in Computer Science*, 92–108. Springer. ISBN 978-3-642-14711-1. URL `http://dx.doi.org/10.1007/978-3-642-14712-8`.

W. Diffie & M.E. Hellman (1976). New directions in cryptography. *IEEE Transactions on Information Theory* **22**(6), 644–654.

D. Eastlake & P. Jones (2001). RFC 3174: Secure Hash Algorithm 1 (SHA1).

Taher El-Gamal (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* **31**(4), 469–472.

David Freeman (2010). Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In *Advances in Cryptology – EUROCRYPT 2010*, Henri Gilbert, editor, volume 6110 of *Lecture Notes in Computer Science*, 44–61. Springer Berlin / Heidelberg. URL http://dx.doi.org/10.1007/978-3-642-13190-5_3. 10.1007/978-3-642-13190-5_3.

Steven Galbraith, Keith Harrison & David Soldera (2002). Implementing the Tate Pairing. In *Algorithmic Number Theory*, Claus Fieker & David Kohel, editors, volume 2369 of *Lecture Notes in Computer Science*, 69–86. Springer Berlin / Heidelberg. URL http://dx.doi.org/10.1007/3-540-45455-1_26. 10.1007/3-540-45455-1_26.

Shafi Goldwasser, Silvio Micali & Ronald L. Rivest (1988). A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing* **17**(2), 281–308. ISSN 0097-5397.

Kumar Murty Ian Blake & Guangwu Xu (2004). Refinements of Miller's Algorithm for Computing Weil/Tate Pairing http://eprint.iacr.org/.

INDICARE (2005). The INformed DIalogue about Consumer Acceptability of DRM Solutions in Europe (INDICARE). URL http://www.indicare.org/survey. [Online; accessed 13-April-2011].

Markus Jakobsson, Kazue Sako & Russell Impagliazzo (1996). Designated Verifier Proofs and Their Applications. In *Advances in Cryptology - EUROCRYPT 96*, Ueli Maurer, editor, volume 1070 of *Lecture Notes in Computer Science*, 143–154. Springer Berlin / Heidelberg. URL http://dx.doi.org/10.1007/3-540-68339-9_13. 10.1007/3-540-68339-9_13.

Antoine Joux (2002). The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In *Algorithmic Number Theory*, Claus Fieker & David Kohel, editors, volume 2369 of *Lecture Notes in Computer Science*, 11–18. Springer Berlin / Heidelberg. URL http://dx.doi.org/10.1007/3-540-45455-1_3. 10.1007/3-540-45455-1_3.

Antoine Joux (2004). A One Round Protocol for Tripartite Diffie–Hellman. *Journal of Cryptology* **17**, 263–276. ISSN 0933-2790. URL http://dx.doi.org/10.1007/s00145-004-0312-y. 10.1007/s00145-004-0312-y.

R.H. Koenen, J. Lacy, M. MacKay & S. Mitchell (2004). The long march to interoperable digital rights management. *Proceedings of the IEEE* **92**(6), 883–897.

David W. Kravitz & Thomas S. Messerges (2005). Achieving media portability through local content translation and end-to-end rights management. In *DRM '05:*

Proceedings of the 5th ACM workshop on Digital rights management, 27–36. ACM, New York, NY, USA. ISBN 1-59593-230-5.

Sébastien Kunz-Jacques & David Pointcheval (2006). About the Security of MTI/C0 and MQV. In *SCN*, Roberto De Prisco & Moti Yung, editors, volume 4116 of *Lecture Notes in Computer Science*, 156–172. Springer. ISBN 3-540-38080-9. URL http://dx.doi.org/10.1007/11832072_11.

Fabien Laguillaumie & Damien Vergnaud (2004). Designated Verifier Signatures: Anonymity and Efficient Construction from Any Bilinear Map. In *SCN 2004*, Carlo Blundo & Stelvio Cimato, editors, volume 3352 of *Lecture Notes in Computer Science*, 105–119. Springer. ISBN 3-540-24301-1.

Fabien Laguillaumie & Damien Vergnaud (2007). Multi-designated verifiers signatures: anonymity without encryption. *Information Processing Letters* **102**(2-3), 127–132. ISSN 0020-0190.

Benoît Libert & Damien Vergnaud (2008a). Multi-use unidirectional proxy resignatures. In *ACM Conference on Computer and Communications Security*, Peng Ning, Paul F. Syverson & Somesh Jha, editors, 511–520. ACM. ISBN 978-1-59593-810-7.

Benoît Libert & Damien Vergnaud (2008b). Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In *Public Key Cryptography – PKC 2008*, Ronald Cramer, editor, volume 4939 of *Lecture Notes in Computer Science*, 360–379. Springer Berlin / Heidelberg. ISBN 978-3-540-78439-5. URL http://dx.doi.org/10.1007/978-3-540-78440-1_21. 10.1007/978-3-540-78440-1_21.

Helger Lipmaa (2010). Random Oracle Model Link Farm. http://research.cyber.ee/~lipmaa/crypto/link/rom/. [Online; accessed 13-April-2011].

Dennis Meffert (2009). *Bilinear Pairings in Cryptography*. Master's thesis, Radbound University Nijmegen. URL http://www.math.ru.nl/~bosma/Students/MScThesis_DennisMeffert.pdf.

V. Miller (1986). Short programs for functions on curves. *Unpublished manuscript* **97**, 101–102.

Michael Nüsken (2010). Tate Pairing, An extract of Elliptic Curve Cryptography. *Unpublished manuscript* .

PACE-Project (2008). Pairings and Advances in Cryptology for E-cash (PACE) project. URL https://pace.rd.francetelecom.com. [Online; accessed 13-April-2011].

R. Rivest (1992). RFC1321: The MD5 message-digest algorithm .

Shahrokh Saeednia, Steve Kremer & Olivier Markowitch (2003). An Efficient Strong Designated Verifier Signature Scheme. In *ICISC 2003*, Jong In Lim & Dong Hoon Lee, editors, volume 2971 of *Lecture Notes in Computer Science*, 40–54. Springer. ISBN 3-540-21376-7.

Jun Shao (2009). Bibliography on Proxy Re-Cryptography. `http://ndc.zjgsu.edu.cn/~jshao/prcbib.htm`. [Online; accessed 13-April-2011].

Jun Shao, Min Feng, Bin Zhu, Zhenfu Cao & Peng Liu (2010). The security model of unidirectional proxy re-signature with private re-signature key. In *Proceedings of the 15th Australasian conference on Information security and privacy*, ACISP'10, 216–232. Springer-Verlag, Berlin, Heidelberg. ISBN 3-642-14080-7, 978-3-642-14080-8. URL `http://portal.acm.org/citation.cfm?id=1926211.1926228`.

Ron Steinfeld, Laurence Bull, Huaxiong Wang & Josef Piperzyk (2003). Universal Designated-Verifier Signatures. In *ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, 523–542. Springer. ISBN 3-540-20592-6.

NR Sunitha & BB Amberker (2009). Multi-use unidirectional forward-secure proxy re-signature scheme. In *Proceedings of the 3rd IEEE international conference on Internet multimedia services architecture and applications*, 223–228. IEEE Press.

Gelareh Taban, Alvaro A. Cárdenas & Virgil D. Gligor (2006). Towards a secure and interoperable DRM architecture. In *DRM '06: Proceedings of the ACM workshop on Digital rights management*, 69–78. ACM, New York, NY, USA. ISBN 1-59593-555.

T.C.G. (2008). Trusted Computing Group. URL `http://www.trustedcomputinggroup.org`. [Online; accessed 13-April-2011].

U.S. Department of Commerce / National Institute of Standards and Technology (2000). FIBS PUB 186-2, Digital Signature Standard (DSS). *Federal Information Processing Standards Publication* .

Lawrence C. Washington (2008). *Elliptic Curves: Number Theory and Cryptography, Second Edition*. Chapman & Hall/CRC. ISBN 9781420071467.

Brent Waters (2005). Efficient Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology – EUROCRYPT 2005*, Ronald Cramer, editor, volume 3494 of *Lecture Notes in Computer Science*, 114–127. Springer Berlin / Heidelberg. ISBN 3-540-25910-4.

Annette Werner (2002). *Elliptische Kurven in der Kryptographie*. Springer, Berlin. ISBN 978-3540425182.

T. Jonas Özgan