Revisiting the security model for aggregate signature schemes

by

Marie-Sarah Lacharité

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2014

Author's Declaration

Abstract

Aggregate signature schemes combine the digital signatures of multiple users on different messages into one single signature. The Boneh-Gentry-Lynn-Shacham (BGLS) aggregate signature scheme is one such scheme, based on pairings, where anyone can aggregate the signatures in any order. We suggest improvements to its current chosen-key security model. In particular, we argue that the scheme should be resistant to attackers that can adaptively choose their target users, and either replace other users' public keys or expose other users' private keys. We compare these new types of forgers to the original targeted-user forger, building up to the stronger replacement-and-exposure forger. Finally, we present a security reduction for a variant of the BGLS aggregate signature scheme with respect to this new notion of forgery. Recent attacks by Joux and others on the discrete logarithm problem in small-characteristic finite fields dramatically reduced the security of many type I pairings. Therefore, we explore security reductions for BGLS with type III rather than type I pairings. Although our reductions are specific to BGLS, we believe that other aggregate signature schemes could benefit from similar changes to their security models.

Acknowledgements

Thank you to my supervisor, Alfred Menezes, for his excellent guidance as I wrote this thesis. I appreciate his encouragement and patience. I am also thankful to Edlyn Teske-Wilson and David Jao, the other members of my Reading Committee, for their insightful comments.

I am also grateful for the financial support of the Department of Combinatorics and Optimization and the Ontario Ministry of Training, Colleges, and Universities, in the form of an Ontario Graduate Scholarship.

Table of Contents

List of Figures

Chapter 1

Introduction

Reductions give us confidence in the security of cryptographic schemes, but they are not simple to interpret. The tightness of a reduction from solving a primitive to breaking a protocol indicates how much of the primitive's hardness is inherited by the protocol. If the reduction is not tight, then its security guarantee is weak: breaking the protocol takes only some fraction of the work required to solve the primitive. This thesis examines two aspects of security reductions: tightness of the reduction and what it means for an adversary to break the scheme. We examine not only reductions from solving primitives to breaking protocols, but also reductions among different ways of breaking protocols.

Good security definitions are important—they specify what capabilities attackers have and what they must accomplish to break a protocol. The best security definitions typically assume that adversaries have strong capabilities and weak goals. For instance, a secure digital signature scheme must be existentially unforgeable under adaptive chosen-message attack, a secure message authentication code (MAC) scheme must be existentially unforgeable under adaptive chosen-message attack, and a secure public-key encryption scheme must be indistinguishable under adaptive chosen-ciphertext attack. These definitions are all in the single-user setting: we assume that only one user is signing messages, only one pair of users is tagging messages, and only one user is receiving encrypted messages. In the multi-user setting, security definitions become more complex.

In this thesis, we explore types of attackers for a scheme that is naturally in the multi-user setting—the Boneh-Gentry-Lynn-Shacham (BGLS) aggregate signature scheme. Its security is based on solving the modified computational co-Diffie-Hellman (co-CDH*) problem in the domain groups of a pairing. In the original security model, an attacker receives one public key to target and can choose the public keys of any other users in its forged signature. Most signature schemes have security models where attackers do not choose their target users. We believe these security models are not as strong as they could be.

The thesis is organized as follows. In Chapter 2, we review digital signature schemes, elliptic curves, and pairings, which act on groups of points on elliptic curves. We justify our decision to consider only type III pairings and define some Diffie-Hellman problems, including the co-CDH* problem that is the primitive for the BGLS aggregate signature scheme. The chapter ends with observations about why the tightness of a reduction is important.

Chapter 3 reviews the BLS signature scheme and BGLS aggregate signature scheme. We discuss the authors' requirement for distinct messages in an aggregate signature and then present the original BGLS security reduction.

In Chapter 4, we examine existing security definitions for other aggregate signature schemes and related schemes, such as multi-signatures. Then, we begin exploring other types of BGLS forgers. First, we examine aggregate forgers that can choose their target users and expose other users' private keys. Next, we examine aggregate forgers that can choose their target users and replace the public keys of any other users. The chapter concludes with a section comparing these two types of attackers.

In Chapter 5, we present our new security definition, based on resistance to a forger with the combined capabilities of the exposure and replacement forgers from the previous chapter. We present a security reduction for BGLS aggregate signatures with respect to this type of forgery.

1.1 Notation and useful mathematical results

In this section, we state two useful results and summarize our notation.

First,

$$\arg\max_x \{x(1-x)^{n-1}\} = \frac{1}{n}. \tag{1.1}$$

We use this optimal value of x to maximize success probabilities in reductions. The derivative

$$\frac{d}{dx}\left(x(1-x)^{n-1}\right) = (1-x)^{n-2}(1-x-(n-1)x)$$

equals 0 when $x = 1$ or $1 = nx$.

Second,

$$e^x = \lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n. \tag{1.2}$$

In particular, $e^{-1} = \lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n$, and this limit converges rapidly. We use this approximation of powers of e when analyzing the success probability of reductions.

Finally, we briefly list some of our notation.

$[\mathbf{n}]$ is the set of positive integers $\{1, \ldots, n\}$.

$\mathbf{a} \in_{\mathbf{R}} \mathbf{B}$ means that the element a is chosen uniformly and randomly from the set B.

$\mathbf{h}(\cdot)$ is a hash function.

H is a hashing oracle.

$\mathbf{S_i}$ is a signing oracle for user i.

$\mathbf{e}(\cdot,\cdot)$ is a pairing.

e is Euler's number, the base of the natural logarithm.

P and Q are probabilities.

p and q are primes.

$\mathbf{1_G}$ is the identity of the group G.

$\langle \mathbf{g} \rangle$ is the group generated by g.

$\mathbf{T_m}$ **and** $\mathbf{T_e}$ are the times required to perform multiplication and exponentiation in a given group or groups.

$(\mathbf{x},\mathbf{y}) = (\mathbf{g_1}^{\mathbf{z}},\mathbf{g_2}^{\mathbf{z}})$ is a public key in the BLS or BGLS signature schemes with type III pairings. The corresponding private key is z.

$(\mathbf{u},\mathbf{v}) = (\mathbf{g_1}^{\mathbf{w}},\mathbf{g_2}^{\mathbf{w}})$ is a public key that was chosen by a forger in the BLS scheme with type III pairings, or replaced by a forger in the BGLS scheme with type III pairings. The corresponding private key is w.

$(\mathbf{x}',\mathbf{y}') = (\mathbf{g_1}^{\mathbf{z}'},\mathbf{g_2}^{\mathbf{z}'})$ is a public key that may have been modified in some way. For instance, it could represent a public key after interacting with a forger that can replace keys. It could also represent a key created by one forger, possibly as a function of a key it received, to give to another forger. The corresponding private key is z'.

Chapter 2

Background

Digital signatures are ubiquitous online. Every time an SSL connection is established between a client and a server, the client verifies the server's identity by verifying its certificate—it checks the validity of a signature by a certificate authority on the server's identity and public key.

2.1 Digital signatures

In this section, we introduce digital signature schemes and what it means for them to be secure, using RSA-FDH signatures as an example. The security reduction for RSA-FDH signatures is very similar to the security reduction for BLS signatures, which form the basis of the BGLS aggregate signature scheme.

A digital signature, like its written equivalent, verifies the origin of a message or indicates approval of a document. Anyone can verify the authenticity of a signature on paper, but only one person can create it. Diffie and Hellman proposed the first digital replacement, based on an abstract public-key cryptosystem constructed from a trapdoor function [15]. A trapdoor function is one that is easy to compute but hard to invert without knowledge of the trapdoor information. A trapdoor function can be the encryption function of a public-key encryption scheme, while the trapdoor information is the private key, which allows a user to decrypt messages. A signature scheme naturally arises from such an encryption scheme. To sign a message, a user decrypts it with its private key. To verify a signature, the receiver encrypts it with the sender's public key. This scheme was the first to provide a "purely digital, unforgeable, message dependent signature" [15].

Diffie and Hellman's description of digital signatures, or "one-way authentication," was only a concept, but concrete schemes soon followed. Rivest, Shamir, and Adleman proposed RSA signatures at the same time as the RSA cryptosystem, in 1978 [29]. In the RSA cryptosystem and signature scheme, the modulus n is the product of two primes and the integers e and d are inverses of each other modulo $\phi(n)$, where $\phi(\cdot)$ denotes the Euler phi function. Messages are integers modulo n. The trapdoor function is the RSA function, exponentiation by e modulo n. The trapdoor information that allows inverting this function is d, the private key. To encrypt a message, the sender raises it to the power of the receiver's public key e. To decrypt a message, the receiver raises it to the power of its own private key d. Since $m^{ed} \equiv m \pmod{n}$, the receiver obtains the message. To sign a message, the signer raises it to the power of its private key d. To verify the signature on a message, any user can raise it to the power of the purported signer's public key e. Again, since $m^{ed} \equiv m \pmod{n}$, the user will obtain the message if the signature is valid.

The inventors of RSA stated that the security of the RSA cryptosystem and digital signature scheme "rests in part on the difficulty of factoring the published divisor, n." They felt "reasonably confident that [computing e-th roots modulo n without factoring n] is computationally intractable." Today, we call this problem the "RSA problem" or the eth-root problem, and solving it corresponds to forging a signature on a message or decrypting a message.

Ten years after the proposal of RSA signatures, Goldwasser, Micali, and Rivest formalized the notions of a digital signature scheme and what it means to break such a scheme [19]. We present a simplified version of their definition.

Definition 2.1. A **digital signature scheme** has the following components:

- A message space, key space, and signature space.
- A public, randomized **key generation** algorithm that receives a security parameter and returns a public-private key pair in the key space for that security parameter.
- A **signing** algorithm that receives a message and a user's private key, and returns a signature by that user on that message.
- A public **verification** algorithm that receives a signature, a message, and a user's public key, and outputs TRUE if the signature is valid for the message by that user, or FALSE otherwise.

A digital signature scheme is correct if the verification algorithm returns TRUE for any signature obtained from the signing algorithm. In a digital signature scheme, each user that wants to sign messages must have a public-private key pair.

Goldwasser, Micali, and Rivest also identified what it means to break a digital signature scheme [19].

Definition 2.2. A digital signature scheme is **secure** if it is resistant to existential forgery under adaptive chosen-message attack. A signature scheme is $(\mathbf{t}, \epsilon, \mathbf{q_h}, \mathbf{q_s})$**-secure** against existential forgery under adaptive chosen message attack if there is no adversary that breaks the scheme in time at most t with probability at least ϵ and makes at most q_h hashing queries and q_s signing queries.

That is, no attacker that is given a public key can forge a single signature on any new message given access to an oracle that signs messages of its choice. The attacker may choose which messages to give to the signing oracle based on its previous responses.

This definition of security is strong because the adversary is powerful and it has a weak goal: it can mount a chosen-message attack and all it must do is forge a signature on any message of its choice. An attacker could have a stronger goal, such as selective forgery, universal forgery, or recovery of the private key. It could have fewer capabilities, such as receiving only some message-signature pairs, or having to choose which messages it will ask to be signed before seeing the user's public key. However, the strongest notion of security is against an attacker with the weakest goal and greatest capabilities.

The RSA signature scheme, as we described it earlier, is not resistant to existential forgery under chosen-message attack. An attacker can select an arbitrary signature σ modulo n and compute its corresponding message $m = \sigma^e \mod n$. Then, σ is a valid forged signature on m. Efficiency is another problem: a bigger modulus is required to sign longer messages. One solution to these problems is to sign the hash of a message.

In 1996, Bellare and Rogaway proved that any signature scheme based on a trapdoor permutation, such as the RSA function, is secure when message hashes are signed, provided the hash function is random and uniformly maps messages onto the domain of the signing function [7]. A trapdoor permutation is a bijective trapdoor function whose range is a permutation of its domain. In particular, the RSA signature scheme with a full-domain uniform hash function (RSA-FDH) is secure in the random oracle model.

The use of digital signatures in practice provokes many questions, such as how to combine multiple digital signatures to reduce their size or the verification time. How can we efficiently combine the signatures of many users on the same message or on many messages? Multi-signature schemes solve the first problem. Aggregate signature schemes such as BGLS, which we focus on, solve the second problem. Before introducing BLS signatures and BGLS aggregate signatures, we review the basics of elliptic curves and pairings.

2.2 Elliptic curves and pairings

An elliptic curve is a mathematical object often used in cryptography because its points form a group. In general, a curve is the set of points with coordinates in a certain field that satisfy an equation with coefficients in the same field. In this section, we first define elliptic curves as types of plane curves. Then, we transform projective coordinates to affine coordinates and present the reduced Weierstrass form for elliptic curves.

Definition 2.3. For any field $\mathbb{K}$, the **projective plane over** $\mathbb{K}$, $P^2(\mathbb{K})$, is the set of equivalence classes of the relation $\sim$ on non-zero points in $\mathbb{K}^3$, where $(a_1, a_2, a_3) \sim (b_1, b_2, b_3)$ if there exists an element x in $\mathbb{K}$ such that $a_i = xb_i$ for $i = 1$, 2, and 3.

We denote the equivalence class containing (a, b, c) by $(a : b : c)$ and call it a projective point. Let $\overline{\mathbb{K}}$ denote the algebraic closure of $\mathbb{K}$ and let $\mathbb{L}$ be any extension field of $\mathbb{K}$—any field such that $\mathbb{K} \subseteq \mathbb{L} \subseteq \overline{\mathbb{K}}$. Next, we define a type of curve whose points are in the projective plane $P^2(\overline{\mathbb{K}})$.

Definition 2.4.1. A **non-singular plane curve $\mathcal{C}$ of degree d over** $\mathbb{K}$ is a curve defined by a homogeneous degree-d polynomial f in three variables, say x, y, and z, with coefficients in $\mathbb{K}$, such that no point in $P^2(\overline{\mathbb{K}})$ on the curve is a solution to $\frac{\partial}{\partial x}f = \frac{\partial}{\partial y}f = \frac{\partial}{\partial z}f = 0$. The set of points on $\mathcal{C}$ are all of the points $(x_0 : y_0 : z_0)$ in $P^2(\overline{\mathbb{K}})$ such that $f(x_0, y_0, z_0) = 0$.

Definition 2.4.2. The set of $\mathbb{L}$-rational points $\mathcal{C}(\mathbb{L})$ on $\mathcal{C}$ comprises all points $(x_0 : y_0 : z_0)$ in $P^2(\mathbb{L})$ such that $f(x_0, y_0, z_0) = 0$.

The points where the partial derivatives in Definition 2.4.1 simultaneously vanish are singular points. We avoid them because they do not have well-defined tangent lines. Therefore, a non-singular curve is also called a smooth curve.

Definition 2.5. An **elliptic curve over** $\mathbb{K}$ is a cubic, non-singular plane curve over $\mathbb{K}$, with a $\mathbb{K}$-rational point on that curve.

Every elliptic curve over $\mathbb{K}$ is isomorphic to a curve in (projective) general Weierstrass form:

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

where a_1, a_2, a_3, a_4, and a_6 are elements of $\mathbb{K}$ and the point $\mathcal{O}$, called the point at infinity, corresponds to $(0 : 1 : 0)$.

Although this thesis does not consider the problem of representing points on elliptic curves, we present the simplified general Weierstrass form by converting projective coordinates to affine coordinates. The projective point $(a : b : c)$ is the set of all points $(a\lambda, b\lambda, c\lambda)$, where λ is any non-zero element in the field $\mathbb{K}$. If we set $\lambda = c^{-1}$, we can associate any projective point $(a : b : c)$ with an affine point $(a', b') = (ac^{-1}, bc^{-1})$. We simply denote the point at infinity, $(0 : 1 : 0)$, by $\mathcal{O}$. Hence, we obtain the following alternate definition of an elliptic curve.

Definition 2.6.1. An **elliptic curve $\mathcal{E}$ over** $\mathbb{K}$ is the set of all non-zero points in $\overline{\mathbb{K}}^2$ satisfying the non-singular equation $Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$, where a_1, a_2, a_3, a_4, and a_6 are elements of $\mathbb{K}$, together with the point at infinity. Non-singularity requires that the partial derivatives do not simultaneously vanish at any point in $\overline{\mathbb{K}}^2$ that is on the curve.

Definition 2.6.2. The set of $\mathbb{L}$-rational points $\mathcal{E}(\mathbb{L})$ on $\mathcal{E}$ comprises the point at infinity and all points (x_0, y_0) in $\mathbb{L}^2$ that satisfy the curve's affine general Weierstrauss equation.

For any extension field $\mathbb{L}$ of $\mathbb{K}$, the set of $\mathbb{L}$-rational points on an elliptic curve form an abelian group with point addition as the group operation. The point at infinity is the group identity—adding any point to it results in that point. We present only a brief description of how to geometrically construct the sum of two affine points. First, construct a line through the two points, or a tangent line if doubling a point. This line intersects the curve at exactly one other point. Reflect this third point about the x-axis, i.e., negate its y-coordinate, to get the "sum" of the first two points. If the line through two points is vertical, then the third intersection point—their sum—is the point at infinity whose inverse is itself.

The group of points on an elliptic curve is always isomorphic to the product of two cyclic groups. Suppose now that $\mathbb{K}$ and $\mathbb{L}$ are finite fields. The group of $\mathbb{L}$-rational points on an elliptic curve over $\mathbb{K}$ is isomorphic to $\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$ where n_2 divides n_1 and n_2 divides $\#\mathbb{L} - 1$.

There is a special case of the discrete logarithm problem in groups of points on elliptic curves:

Definition 2.7. The **elliptic curve discrete logarithm problem (ECDLP)** in the subgroup $\langle P_1 \rangle$ generated by a point P_1 of order n is to find the integer ℓ in $[0, n-1]$ such that $\ell P_1 = P_2$, given the base point P_1, its order n, and a point P_2 in $\langle P_1 \rangle$.

Elliptic curves are useful in cryptography because this problem is hard: the best-known generic attack on the ECDLP in a group of order n, Pollard's parallelized ρ method, takes time $O(\sqrt{n})$ [30]. When the factorization of n is known, the time is proportional to the square root of n's largest prime factor.

Elliptic curves over finite fields are classified into two types—supersingular and ordinary—depending on whether the characteristic of the field divides a certain quantity relating the order of the field and the number of points on the curve.

Definition 2.8. The **trace of Frobenius** of the elliptic curve $\mathcal{E}(\mathbb{K})$ is $t = \#\mathbb{K} + 1 - \#\mathcal{E}(\mathbb{K})$ where $\#\mathbb{K}$ is the order of the field and $\#\mathcal{E}(\mathbb{K})$ is the number of $\mathbb{K}$-rational points on the elliptic curve $\mathcal{E}$.
If the field's characteristic does divide the trace of Frobenius, then the elliptic curve is **supersingular.** Otherwise, it is **ordinary**.

We examine one final property of elliptic curves over finite fields.

Definition 2.9. Let $\mathcal{E}(\mathbb{K})$ be an elliptic curve and let p be a prime integer that divides $\#\mathcal{E}(\mathbb{K})$ and is co-prime with $\#\mathbb{K}$. The **embedding degree k** of $\mathcal{E}(\mathbb{K})$ with respect to p is the smallest positive integer k such that p divides $(\#\mathbb{K})^k - 1$.

Now that we have briefly examined how groups arise from elliptic curves, we look at pairings. We consider pairings based on the Weil or Tate pairings on elliptic curves over finite fields. We denote the finite field of order q by $\mathbb{F}_q$.

Let G_1, G_2, and G_T be groups of prime order p. The groups G_1 and G_2 can be written multiplicatively or additively since there is only one group of order p up to isomorphism. Although groups of points on elliptic curves have an additive operation, we choose to write all groups multiplicatively. We use the following cryptographic definition of pairings; we do not consider exactly how pairings are constructed over elliptic curves.

Definition 2.10. A **pairing** is a map $e(\cdot, \cdot)$ from $G_1 \times G_2$ to G_T satisfying the following three properties:

(i) **bilinearity.** For all x_1 and x_2 in G_1, and y in G_2, $e(x_1 x_2, y) = e(x_1, y) \cdot e(x_2, y)$. Similarly, for all x in G_1, and y_1 and y_2 in G_2, $e(x, y_1 y_2) = e(x, y_1) \cdot e(x, y_2)$.

(ii) **non-degeneracy.** If $e(x_0, y) = 1_{G_T}$ for all y in G_2, then $x_0 = 1_{G_1}$.
Similarly, if $e(x, y_0) = 1_{G_T}$ for all x in G_1, then $y_0 = 1_{G_2}$.
Synonymously, if x and y are generators of G_1 and G_2, then $e(x, y) \neq 1_{G_T}$.

(iii) **efficiency.** The pairing $e(\cdot, \cdot)$ can be computed in polynomial time in $\log p$, where p is the order of the groups.

By repeatedly applying bilinearity, powers of the operands change into powers of the pairing values and vice versa. For any integers a and b and all group elements $x \in G_1$ and $y \in G_2$,

$$e(x^a, y^b) = e(x^a, y)^b = e(x, y^b)^a = e(x, y)^{ab} = e(x^{ab}, y) = e(x, y^{ab}) = e(x^b, y^a).$$

Galbraith, Paterson, and Smart classified pairings into three types [17]:

Type I: symmetric pairings, where the groups G_1 and G_2 are identical.

Type II: asymmetric pairings, where $G_1 \neq G_2$ and there is a known, efficiently computable isomorphism ψ from G_2 to G_1.

Type III: asymmetric pairings that have no known efficiently computable isomorphism from G_2 to G_1, or from G_1 to G_2.

For most common pairings, G_1 is an order-p subgroup of $\mathcal{E}(\mathbb{F}_q)$, G_2 is an order-p subgroup of $\mathcal{E}(\mathbb{F}_{q^k})$ where k is the embedding degree with respect to a prime divisor p of $\#\mathcal{E}(\mathbb{F}_q)$, and G_T is the order-p subgroup of $\mathbb{F}^*_{q^k}$ [16]. For pairing-based schemes to be secure, the ECDLP in the groups G_1 and G_2 and the DLP in the target group G_T must be hard. A pairing-friendly curve is one that has a large prime-order subgroup and an embedding degree that is big enough so that solving the DLP in G_T is not easy, but small enough so that computing pairing values is not infeasible. Freeman, Scott, and Teske formalized the definition of pairing-friendly curves [16]:

Definition 2.11. An elliptic curve $\mathcal{E}$ over $\mathbb{F}_q$ is **pairing-friendly** if there exists a prime integer $p \geq \sqrt{q}$ dividing $\#\mathcal{E}(\mathbb{F}_q)$ and the embedding degree k of $\mathcal{E}$ with respect to p is less than $(\log_2 p)/8$.

Type I pairings are implemented with supersingular curves over prime fields or fields of characteristic 2 or 3 [17]. Recent work by Joux and others on solving the discrete logarithm problem in fields of small characteristic render these curves, and thus many type I pairings, insecure [3, 20, 21]. Since computations in small-characteristic fields are much more efficient than in prime fields, we choose to ignore type I pairings.

In the paper introducing BLS short signatures, the authors use type II pairings, stating that the isomorphism ψ seems to be necessary for the security reductions [9]. However, Chatterjee, Hankerson, Knapp, and Menezes describe variants of BLS and BGLS signature schemes that use type III pairings, eliminating the need for a known, efficiently computable map ψ [10]. They further argue that type II pairings have no advantage in either performance or security over type III pairings when implementing BLS and BGLS with Barreto-Naehrig pairings. Therefore, in this thesis, we consider only type III pairings.

2.3 Diffie-Hellman problems

Diffie-Hellman problems are the primitives of many cryptographic protocols that involve groups. First, we examine Diffie-Hellman problems that involve elements of either a single group or two groups.

Let g be a generator (any element except the identity) of a multiplicative group G of prime order p. The group G, its order p, and the chosen generator g are public. Let a, b, and c be any three non-zero integers modulo p.

Definition 2.12.1. The **computational Diffie-Hellman problem** (CDH) is to compute g^{ab} when given g^a and g^b.

Definition 2.12.2. The **decisional Diffie-Hellman problem** (DDH) is to determine whether $g^{ab} = g^c$ when given g^a, g^b, and g^c.

A group where solving the CDH problem is hard, but solving the DDH problem is easy is a gap group. We can solve the DDH problem given a type I pairing $e : G \times G \rightarrow G_T$ by checking whether $e(g^a, g^b)$ equals $e(g^c, g)$.

Next, we examine two co-Diffie-Hellman problems that involve two groups, such as the components of the domain of an asymmetric pairing. Again, one of these problems is decisional and one is computational. Let g_1 and g_2 be generators of the multiplicative groups G_1 and G_2, both of prime order p. The groups G_1 and G_2, their order p, and their generators g_1 and g_2 are public. Let a, b, and c be any non-zero integers modulo p.

Definition 2.13.1. The **computational co-Diffie-Hellman problem** (co-CDH) is to compute $h^a \in G_1$ when given ${g_2}^a \in G_2$ and $h \in G_1$.

Definition 2.13.2. The **decisional co-Diffie-Hellman problem** (co-DDH) is to determine whether $h^a = h^c$ when given ${g_2}^a \in G_2$, $h \in G_1$ and $h^c \in G_1$.

The groups used with type II or III pairings are groups where solving the co-CDH problem is hard, but solving the co-DDH problem is easy. Given a type II or III pairing $e : G_1 \times G_2 \rightarrow G_T$, we can solve the co-DDH problem by checking whether $e(h, {g_2}^a)$ equals $e(h^c, g_2)$.

At the end of Section 2.2, we noted that it is possible to modify the BLS and BGLS signature schemes to work with type III pairings [10]. The security reductions for the modified schemes described by Chatterjee et al. require a different computational co-Diffie-Hellman problem, which we denote by co-CDH*.

Definition 2.14. The **modified computational co-Diffie-Hellman problem** (co-CDH*) is to compute $h^a \in G_1$ when given ${g_2}^a \in G_2$, ${g_1}^a \in G_1$, and $h \in G_1$.

This co-CDH* problem is similar to the co-CDH problem, but with one extra piece of information—knowledge of ${g_1}^a$. The co-CDH* problem, therefore, cannot be harder than the co-CDH problem. If an adversary can solve the co-CDH problem, then it can clearly solve the co-CDH* problem.

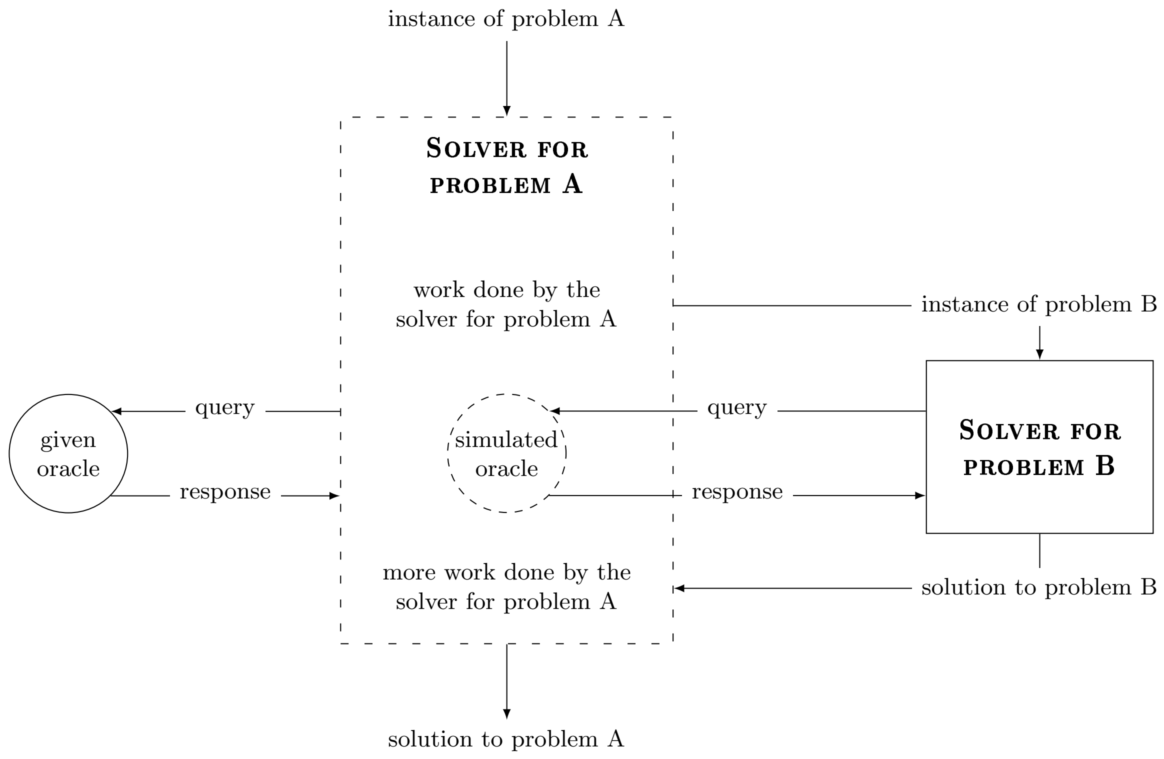


Figure 2.1: We represent reductions with diagrams where dotted lines indicate algorithms or oracles to construct and solid lines represent given algorithms or oracles.

2.4 The importance of tightness

The reduction from solving a primitive to breaking a protocol gives us confidence in a protocol's security. This reduction is an algorithm that can solve the primitive by using a hypothetical subroutine that breaks the protocol and by doing little additional work. "Algorithm" and "solver" refer to deterministic algorithms that have access to a source of random bits.

In this thesis, we augment written descriptions of reductions with diagrams. See Figure 2.1 for a sample reduction from solving problem A to solving problem B. The problems may each access certain oracles. The oracles for problem A are included with the problem instance, whereas the oracles for problem B must be simulated by the solver for problem A. Such a reduction proves, by contraposition, that if solving problem A is hard, then solving problem B is hard. We quantify this hardness by considering the time an algorithm requires and its success probability.

Definition 2.15. An algorithm **$(\mathbf{t}, \epsilon)$-solves problem A** if, given a random instance of problem A, it solves it with probability at least ϵ in time at most t. The probability of success is computed over all possible instances of problem A and all of the solver's coin tosses.

Definition 2.16. Suppose that a reduction uses an algorithm for (t, ϵ)-breaking a protocol to (t', ϵ')-solve a primitive. The **tightness gap** of the reduction is the ratio $(t'/\epsilon')/(t/\epsilon)$. A reduction is **tight** if this ratio is close to 1: when $(t/\epsilon) \approx (t'/\epsilon')$, the protocol inherits the strength of the primitive.

The RSA problem—the problem of computing eth roots modulo n—is the primitive in some security reductions for RSA-FDH. Suppose that the best attack on the RSA problem is factoring the modulus n with the number field sieve. For a 1024-bit modulus n, this attack takes time roughly 2^{80} and succeeds with probability nearly 1. Suppose that the RSA problem is $(2^{70}, 2^{-31})$-hard and that adversaries can compute up to $q_h = 2^{60}$ hashes. The standard reduction from solving the RSA problem to forging an RSA-FDH signature has a tightness gap of q_h. Therefore, this reduction tells us only that RSA-FDH is $(2^{40}, 1/2)$-secure, which is not very assuring. To counter this lack of tightness, we must increase the bitlength of n.

Chatterjee, Menezes, and Sarkar illustrate what a non-tight reduction could mean in the worst case with message authentication code (MAC) schemes, the symmetric-key equivalents of signatures [11]. The best possible attack on an ideal MAC scheme with key length r in the single-user setting is exhaustive key search, which takes time 2^r. The authors present a reduction from breaking a MAC scheme in the single-user setting to breaking it in the multi-user setting. Its tightness gap is n, the number of users. Next, they describe an attack in the multi-user setting that succeeds in time $2^r/n$. The existence of this attack proves that no reduction from single-user MAC to multi-user MAC can be any tighter. Suppose a tighter reduction did exist: given a (t, ϵ)-multi-user MAC forger, it is possible to construct a (t', ϵ')-single-user MAC forger, where $(t'/\epsilon') / (t/\epsilon) = m < n$. Then, breaking single-user MAC takes m times more work than breaking multi-user MAC. However, as noted above, there exists an attack on multi-user MAC that takes time $2^r/n$. Hence, there exists an attack on single-user MAC that takes time $(2^r m)/n < 2^r$, contradicting the fact that the best attack on an ideal MAC scheme takes time 2^r. Therefore, no tighter reduction can exist from breaking single-user MAC to breaking multi-user MAC.

This general approach could apply to other reductions. Consider two problems, A and B. Suppose that the best possible attack on problem A succeeds in time t_A; no attack on problem A can succeed in time faster than t_A. Next, suppose one finds an attack on problem B that succeeds in time t_B. Finally, suppose that there exists a reduction from solving problem A to solving problem B that has a tightness gap of m. Given this attack and the reduction, it is possible to construct an attack on problem A that succeeds in time $m \cdot t_B$. Hence, it must be the case that $m \cdot t_B \geq t_A$, i.e., $m \geq t_A/t_B$. No reduction from solving problem A to solving problem B can have a tightness gap smaller than t_A/t_B.

As the Chatterjee-Menezes-Sarkar example illustrates, a non-tight reduction could indicate the existence of an attack. How should we address non-tight reductions? We could try to find a better reduction with the same primitive. We could weaken the security definition or modify the primitive in a natural way so that the reduction is tighter. We could increase the security parameter size to make up for the tightness gap. In this thesis, we carefully analyze the tightness of all reductions, even those among different types of forgery.

Chapter 3

BLS and BGLS signatures

In this chapter, we review the BLS signature scheme and the BGLS aggregate signature scheme. We restate a proof of the optimality of the BLS security reduction, and explore some constraints on BGLS aggregate signatures.

Our work uses the following assumptions:

- Hash functions are indistinguishable from random functions, so we model them as random oracles.
- When a forger requests a signature on a message from a signing oracle, it has already obtained the hash of this message from the hashing oracle.
- A forger never requests the hash of a message twice, nor a signature from a certain user on the same message twice. (This assumption is without loss of generality for deterministic signature schemes such as BLS and BGLS.)
- When a forger outputs a signature (or aggregate signature) on a message (or messages), every message was previously hashed.
- Signing oracles never output invalid signatures.
- The maximum number of users n in an aggregate signature scheme, or an upper bound on it, is public.

3.1 BLS short signature scheme

The BLS signature scheme has the same security level as the ECDSA signature scheme, but BLS signatures have half the bitlength [9]. The scheme was introduced for type II pairings—those for which an efficiently computable isomorphism from G_2 to G_1 is known. However, we present the modified scheme, due to Chatterjee et al., that also works for type III pairings [10].

Signature Scheme 3.1 (BLS with type III pairing [9, 10]).

- **Set-up:** The groups G_1, G_2, and G_T have prime order p. The groups G_1 and G_2 have generators g_1 and g_2. The function $h(\cdot)$ is a full-domain hash function from $\{0,1\}^*$ to G_1. The map $e(\cdot,\cdot)$ is a type III pairing from $G_1 \times G_2$ to G_T.

- **Key generation:** Let z be a randomly chosen non-zero integer modulo p. The public key is the pair of elements $(x, y) = ({g_1}^z, {g_2}^z)$ in $G_1 \times G_2$. The private key is the integer z.

- **Signing:** To sign a message $m \in \{0,1\}^*$ with the secret key z in $\mathbb{Z}_p^*$, compute the signature $\sigma(m) = h(m)^z$ in G_1.

- **Verification:** To verify the signature σ on a message m by a user with public key (x, y), verify that $e(h(m), y) = e(\sigma, g_2)$.

Given the scheme's parameters and some user's public key, a forger's goal is to compute a valid signature by this user on some message. This problem resembles the co-CDH* problem: given ${g_2}^z$ in G_2, ${g_1}^z$ in G_1 and $h \in G_1$, compute $h^z \in G_1$. This informal reasoning suggests that the security of the BLS signature scheme depends on the hardness of solving the co-CDH* problem in (G_1, G_2).

It is not obvious from the definition of this scheme why the public key must contain both ${g_1}^z$ and ${g_2}^z$, since only the latter is used for verification. The first part of the public key is necessary in the reduction from BLS forgery to solving the modified computational co-Diffie-Hellman problem. The reduction in the opposite direction supports security of the BLS signature scheme with a type III pairing. This reduction, in the following theorem's proof, is depicted in Figure 3.1 on page 15.

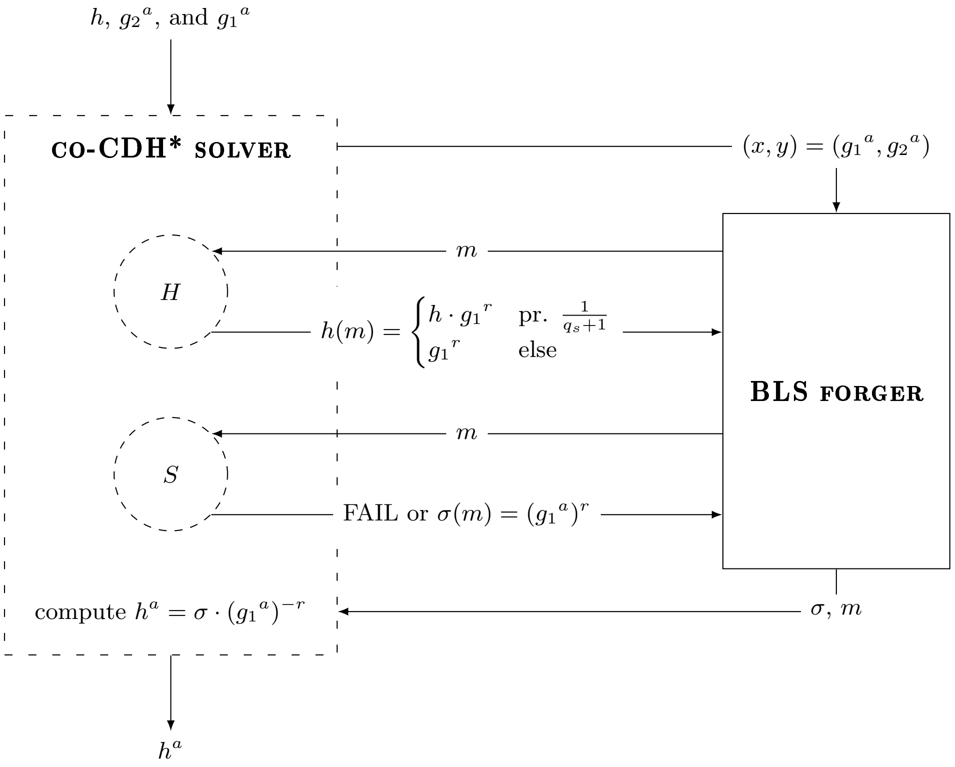
Theorem 3.2 (Security of BLS signature scheme with type III pairing [9, 10]). If solving the co-CDH* problem in (G_1, G_2) is (t', ϵ')-hard, then the BLS signature scheme with a type III pairing is (t, ϵ, q_h, q_s)-secure against existential forgery under adaptive chosen-message attack, for

$$t = t' - (q_h + q_s) \cdot T_e - q_h \cdot T_m, \text{ and}$$
$$\epsilon = \epsilon' \cdot e \cdot (q_s + 1).$$

Proof. We prove the contrapositive of this statement: we build a co-CDH* solver given a forger for BLS. The co-CDH* solver is given $h \in G_1$, ${g_2}^a \in G_2$, and ${g_1}^a \in G_1$. It must somehow use the BLS forger to compute $h^a \in G_1$. The solver must give the forger a public key and simulate hashing and signing oracles for its queries.

First, the solver gives the forger the public key $(x, y) = ({g_1}^a, {g_2}^a)$ in $G_1 \times G_2$. When the forger requests the hash of a message, the solver chooses a random integer $r \in \mathbb{Z}_p$ and returns one of the following elements of G_1:

$$h(m) = \begin{cases} h \cdot {g_1}^r & \text{with probability } P, \\ {g_1}^r & \text{otherwise.} \end{cases}$$



h, ${g_2}^a$, and ${g_1}^a$
co-CDH* solver
$(x, y) = ({g_1}^a, {g_2}^a)$
BLS forger
H
m
$h(m) = \begin{cases} h \cdot {g_1}^r & \text{pr. } \frac{1}{q_s+1} \\ {g_1}^r & \text{else} \end{cases}$
S
m
FAIL or $\sigma(m) = ({g_1}^a)^r$
compute $h^a = \sigma \cdot ({g_1}^a)^{-r}$
σ, m
h^a

Figure 3.1: The reduction from solving the co-CDH problem to BLS forgery has a tightness gap of q_s.*

The solver records the message m and the exponent r. We will determine the optimal probability of P when we compute the solver's success probability.

When the forger requests a signature, the solver's reply depends on the message's hash type. If the message hash is h-dependent, then the solver must abort since it cannot provide a signature. However, if the hash is a random power of g_1, then the solver looks up the appropriate exponent r and returns $\sigma(m) = ({g_1}^a)^r$:

$$\sigma(m) = \begin{cases} \text{FAIL} & \text{if } h(m) \text{ is } h\text{-dependent}, \\ ({g_1}^a)^r & \text{otherwise.} \end{cases}$$

The signature in the latter case is correct:

$$e(\sigma(m), g_2) = e({g_1}^{ar}, g_2) = e({g_1}^r, {g_2}^a) = e(h(m), y).$$

The co-CDH* solver succeeds if and only if the following events occur:

(E_1) The forger does not request a signature on any message with an h-dependent hash. Since the forger makes at most q_s signing queries, this event occurs with probability at least $(1 - P)^{q_s}$.

(E_2) The forger successfully outputs a forgery in time at most t. If it does not request signatures on messages with h-dependent hashes, then the hashing and signing oracles simulated by the co-CDH* solver are indistinguishable from real hashing and signing oracles. Hence, given the first event, this event happens with probability at least ϵ.

(E_3) The forged signature is on a message with an h-dependent hash. The probability of this event given the first two events is at least P.

When these three events occur, the forger outputs a forged signature σ on a message m with hash $h(m) = h \cdot {g_1}^r$. It satisfies $e(\sigma, g_2) = e(h \cdot {g_1}^r, y)$, so the co-CDH* solver computes $h^a = \sigma \cdot ({g_1}^a)^{-r}$. Hence, the probability ϵ' that the co-CDH* solver succeeds is

$$\Pr(E_1 \wedge E_2 \wedge E_3) = \Pr(E_3 \mid E_2 \wedge E_1) \cdot \Pr(E_2 \mid E_1) \cdot \Pr(E_1) \geq P \cdot \epsilon \cdot (1-P)^{q_s}.$$

By Equation (1.1), the value of P that maximizes this lower bound is $P = 1/(q_s + 1)$. Then, applying the approximation for e^{-1} in Equation (1.2) gives the lower bound $\epsilon/(e \cdot (q_s + 1))$.

The time required by the co-CDH* solver is at most $t + (q_h + q_s + 1) \cdot T_e + (q_h + 1) \cdot T_m$. Hence, given a (t, ϵ, q_h, q_s)-forger for BLS, it is possible to build a (t', ϵ')-co-CDH* solver, for

$$t' = t + (q_h + q_s + 1) \cdot T_e + (q_h + 1) \cdot T_m, \text{ and}$$
$$\epsilon' = \frac{\epsilon}{e \cdot (q_s + 1)}. \qquad \square$$

The reduction from solving the RSA problem to forging an RSA-FDH signature is very similar to the reduction from solving the co-CDH* problem to forging a BLS signature. Like the BLS security reduction, it also has a tightness gap of q_s, the number of signature queries the forger can make [13]. In 2002, Coron proved that the RSA-FDH reduction is optimal when the RSA solver uses the forger only once [14]. Kakvi and Kiltz later pointed out that the proof relies on the fact that signatures must be unique, which is not necessarily the case in RSA-FDH if public keys are not certifiable [22]. If the RSA solver gives the RSA-FDH forger a public key (n, e) for which e and $\phi(n)$ are not relatively prime, then signatures are not unique. Determining whether e and $\phi(n)$ are relatively prime is believed to be hard when e is less than $n^{1/4}$. For the BLS scheme, however, signatures are unique for any public key and Coron's result holds, as Knapp noted in 2008 [24]. We state the result here, but omit the proof.

Theorem 3.3 (**Optimality of BLS security reduction** [24]**).** Suppose that a reduction $(t_{\mathcal{R}}, \epsilon_{\mathcal{R}})$-solves the co-CDH* problem by invoking a $(t_{\mathcal{F}}, \epsilon_{\mathcal{F}}, q_h, q_s)$-forger for BLS only once. Then, it is possible to build a $(2(t_{\mathcal{R}} - t_{\mathcal{F}}), \epsilon_{\mathcal{R}} - (\epsilon_{\mathcal{F}}/q_s))$-co-CDH* solver by calling the reduction twice and simulating the forger each time, so no real forger is required at any point.

The proof describes how to build a co-CDH* solver given only a reduction from solving the co-CDH* problem to BLS forgery, but no real forger. Coron's theorem has the following implication. Suppose there exists a new reduction, which uses a forger only once, and proves that if solving the co-CDH* problem is (t', ϵ')-hard, then BLS forgery is (t, ϵ)-hard for some $t' \geq t$ and $\epsilon' = \frac{\epsilon}{q_s} + \delta$. Then, with the construction given in the theorem, we can build an algorithm that solves the co-CDH* problem in time at most $2(t' - t)$ with probability at least

$$\epsilon' - \frac{\epsilon}{q_s} = \left(\frac{\epsilon}{q_s} + \delta\right) - \frac{\epsilon}{q_s} = \delta.$$

If δ is non-negligible, then the existence of this better reduction means that we can solve the co-CDH* problem, which is believed to be hard. Therefore, the existence of better reductions that call the forger only once is unlikely. We emphasize that Coron's theorem does not prove that no tighter reduction exists; it proves only the non-existence of tighter reductions that invoke the BLS forger just once.

3.2 BGLS aggregate signatures

Some applications of digital signatures require many users' valid signatures. Batch verification schemes may verify signatures more efficiently, but they require each signature to be transmitted. For efficiency, we would like to combine these signatures. Multi-signature schemes combine signatures by many users on the same message. Aggregate signature schemes combine many users' signatures on different messages. An aggregate signature scheme is either sequential or general, depending on whether the order of aggregation matters.

In this section, we present the first aggregate signature scheme, BGLS, which is based on BLS signatures [8]. In the following two subsections, we explain why messages in a BGLS aggregate signature must be pairwise distinct, and we present the original security definition. Again, we modify the scheme in the manner of Chatterjee et al. to use type III pairings.

Signature Scheme 3.4 (BGLS with type III pairing [8, 10]).

- **Set-up:** The groups G_1, G_2, and G_T have prime order p. The groups G_1 and G_2 have generators g_1 and g_2. The function $h(\cdot)$ is a full-domain hash function from $\{0,1\}^*$ to G_1. The map $e(\cdot,\cdot)$ is a type III pairing from $G_1 \times G_2$ to G_T.
- **Key generation:** Let z_i be a randomly chosen non-zero integer modulo p. User i's public key is the pair of elements $(x_i, y_i) = ({g_1}^{z_i}, {g_2}^{z_i})$ in $G_1 \times G_2$. The corresponding private key is the integer z_i.
- **Signing:** To sign the k distinct messages $m_1, \dots, m_k \in \{0,1\}^*$ with secret keys $z_1, \dots, z_k \in \mathbb{Z}_p$, compute the aggregate signature $\sigma_A = \prod_{i=1}^{k} h(m_i)^{z_i}$ in G_1.
- **Verification:** To verify the aggregate signature σ_A on messages $m_1, \dots, m_k$ by users with public keys $(x_1, y_1), \dots, (x_k, y_k)$, verify that the messages are pairwise distinct and $\prod_{i=1}^{k} e(h(m_i), y_i) = e(\sigma_A, g_2)$.

Aggregation can be performed by anyone and the resulting signature has the same size as a single BLS signature. Verification succeeds when each individual signature is valid:

$$\prod_{i=1}^{k} e\left(h(m_i), y_i\right) = \prod_{i=1}^{k} e\left(h(m_i), {g_2}^{z_i}\right) = e\left(\prod_{i=1}^{k} h(m_i)^{z_i}, g_2\right) = e\left(\sigma_A, g_2\right).$$

3.2.1 Why should messages be distinct?

The BGLS aggregate signature scheme requires that all messages be distinct, otherwise BGLS is vulnerable to the following rogue key attack. Suppose honest user 1 has public key (x_1, y_1). A malicious user picks a random integer z modulo p and publishes $(x_2, y_2) = ({x_1}^{-1}{g_1}^z, {y_1}^{-1}{g_2}^z)$ as its public key. Then, the attacker can compute a signature on any message m and claim that it was signed by both itself and the first user—it simply computes $\sigma_A = h(m)^z$. This signature is valid since

$$e\left(h(m), y_1\right) \cdot e\left(h(m), y_2\right) = e\left(h(m), y_1({y_1}^{-1}{g_2}^z)\right) = e\left(h(m), {g_2}^z\right) = e\left(\sigma_A, g_2\right).$$

The creators of BGLS were aware of this attack and suggested the following three countermeasures [8]:

1. Require users to prove knowledge of their private keys.
   - Users could disclose their private keys to a trusted party.
   - Users could prove knowledge of their private keys with zero-knowledge proofs.
2. Require users to prove possession of their private keys.
   - Users could sign their certificate request message.
   - Users could sign random messages that will never be used in practice.
3. Require all of the messages in one aggregate signature to be distinct.

The authors suggest that the third option might be the simplest: a user could prepend its public key to a message, creating an "enhanced message," before hashing it. Bellare, Namprempre, and Neven argue that hashing enhanced messages reduces the problem but does not eliminate it [4]. They point out that in some settings, aggregate signatures could genuinely include multiple signatures by the same user on the same message. For example, this situation could occur when aggregation is used to store many digital signatures. They provide a security reduction for the case of enhanced messages. They also present a tight security reduction for a modification of BGLS where each signer prepends a random bit to the enhanced message before signing it. We use a similar technique in Section 5.1 to give a security reduction for BGLS with respect to stronger adversaries.

In this thesis, we simply require that all messages in an aggregate signature be distinct. Our reductions are in the plain public-key model: any valid public key can be certified. In this model, suggested by Bellare and Neven, there is no requirement for proof of knowledge or possession of the private key [5]. In the BGLS scheme, a valid public key is one that

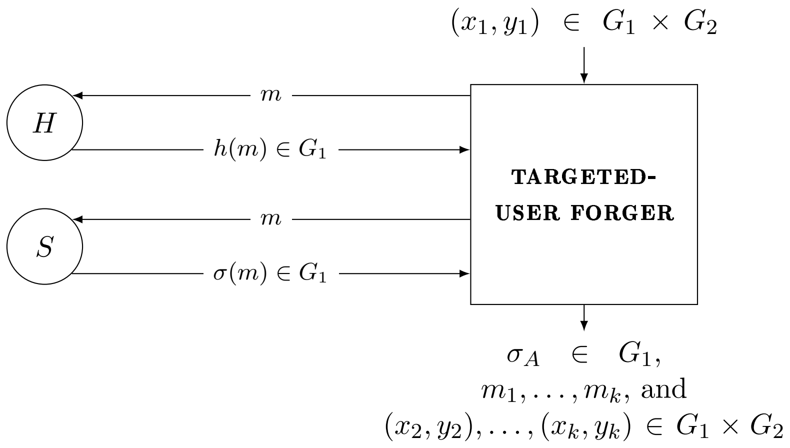


Figure 3.2: Capabilities and goals of a targeted-user forger.

has the form (x, y) where the discrete logarithm of x with respect to g_1 equals the discrete logarithm of y with respect to g_2. Validity of a public key can be verified by checking that $e(g_1, y) = e(x, g_2)$. When a user registers with a certificate authority, it does not have to provide evidence of knowing its own private key.

3.2.2 Original security definition

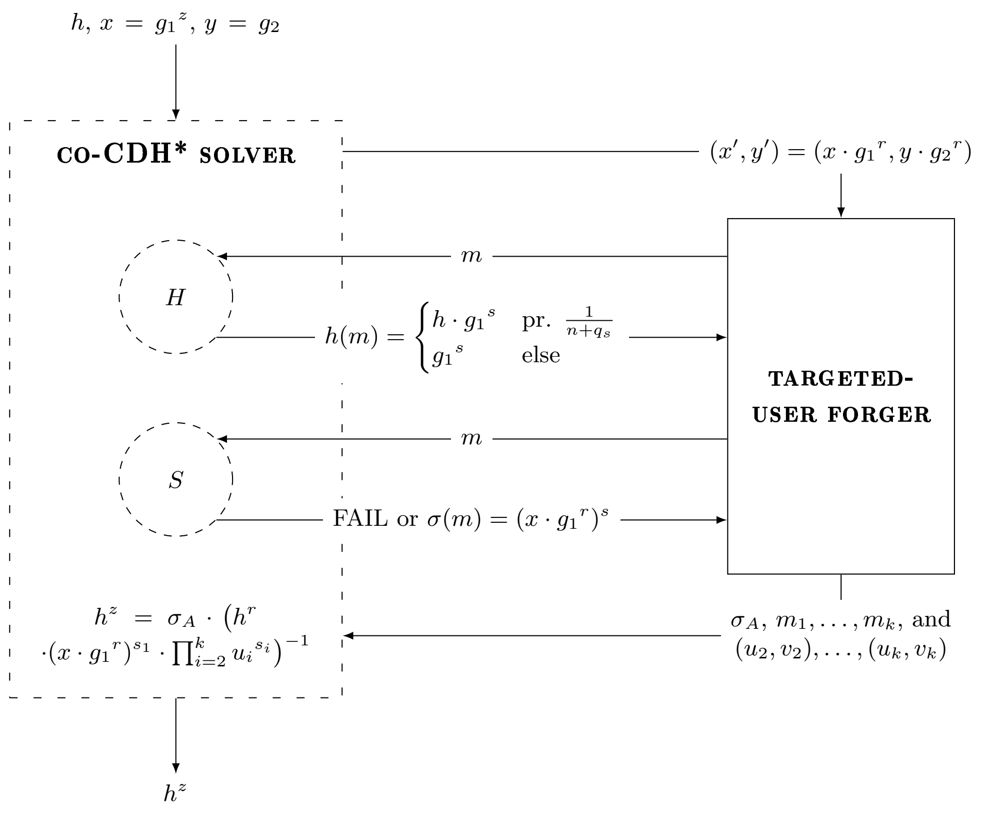
The first security definition for a general aggregate signature scheme was introduced with BGLS [8]. In this section, we restate this definition of what it means for an attacker to break an aggregate signature scheme. Instead of calling this attack "existential forgery in the aggregate chosen-key model," we call it "targeted-user forgery" to emphasize that the goal is existential forgery under chosen-message attack for a particular user.

Let $e(\cdot, \cdot)$ be a type III pairing from $G_1 \times G_2$ to G_T. Consider an instance of BGLS with at most n users, where user i has public key (x_i, y_i) and private key z_i.

Definition 3.1.1. A **targeted-user forger** has the following capabilities and goals. It is given a randomly chosen public key (x_1, y_1) in $G_1 \times G_2$. It adaptively queries a hashing oracle and a signing oracle with messages of its choice.
For some positive integer k that is at most n, the forger must output $k-1$ public keys of its choice $(x_2, y_2), \ldots, (x_k, y_k)$, k distinct messages $m_1, \ldots, m_k$, and a valid aggregate signature σ_A comprising user i's signature on message m_i, for each i from 1 to k. The forger succeeds if it never requested the first user's signature on m_1.

Definition 3.1.2. A $(\mathbf{t}, \epsilon, \mathbf{q_h}, \mathbf{q_s})$**-targeted-user forger** makes at most q_h hashing queries, at most q_s signing queries, runs in time at most t, and succeeds with probability at least ϵ. The success probability is computed over all possible inputs (x_1, y_1) in $G_1 \times G_2$ and all of the forger's coin tosses.

Definition 3.1.3. An aggregate signature scheme is $(\mathbf{t}, \epsilon, \mathbf{q_h}, \mathbf{q_s})$**-secure against targeted-user forgery** if no (t, ϵ, q_h, q_s)-targeted-user forger exists.



$h,\ x = g_1{}^z,\ y = g_2$
co-CDH* solver
$(x', y') = (x \cdot g_1{}^r, y \cdot g_2{}^r)$
H
m
$h(m) = \begin{cases} h \cdot g_1{}^s & \text{pr. } \frac{1}{n+q_s} \\ g_1{}^s & \text{else} \end{cases}$
TARGETED-USER FORGER
S
m
FAIL or $\sigma(m) = (x \cdot g_1{}^r)^s$
$h^z = \sigma_A \cdot (h^r \cdot (x \cdot g_1{}^r)^{s_1} \cdot \prod_{i=2}^k u_i{}^{s_i})^{-1}$
$\sigma_A, m_1, \ldots, m_k$, and $(u_2, v_2), \ldots, (u_k, v_k)$
h^z

Figure 3.3: The reduction from solving the co-CDH problem to targeted-user forgery has a tightness gap of $n + q_s$.*

The original security reduction for BGLS with respect to this type of forgery has a tightness gap of $q_s + n$. We represent the reduction in the proof of Theorem 3.5 in Figure 3.3 on page 20.

Theorem 3.5 (**Security of BGLS aggregate signature scheme with type III pairing** [8]). If solving the co-CDH* problem in (G_1, G_2) is (t', ϵ')-hard, then the BGLS aggregate signature scheme with a type III pairing $e : G_1 \times G_2 \to G_T$ is (t, ϵ, q_h, q_s)-secure against targeted-user forgery, for

$$t = t' - (q_h + q_s + n + 3) \cdot T_e - (q_h + n + 2) \cdot T_m, \text{ and}$$
$$\epsilon = \epsilon' \cdot e \cdot (n + q_s).$$

Proof. We prove the contrapositive of this statement: we show how to build a co-CDH* solver given a targeted-user forger for BGLS. The solver receives an instance of the co-CDH* problem, say h and $x = g_1{}^z$ in G_1, and $y = g_2{}^z$ in G_2. It must eventually output h^z in G_1.

First, it gives the targeted-user forger the public key $(x', y') = (x \cdot g_1{}^r, y \cdot g_2{}^r)$ where r is a randomly chosen integer modulo p. When the targeted-user forger requests the hash of a message, the solver computes a random power of g_1 and multiplies it with h with some probability:

$$h(m) = \begin{cases} h \cdot {g_1}^s & \text{with probability } P, \\ {g_1}^s & \text{otherwise.} \end{cases}$$

The solver records (m, s), where s is a randomly chosen integer modulo p, and whether the hash depends on h. We will determine the optimal value of P when computing the solver's success probability.

If the targeted-user forger requests a signature on a message whose hash is h-dependent, then the solver fails. Otherwise, say $h(m) = {g_1}^s$, the solver gives the targeted-user forger $\sigma(m) = (x \cdot {g_1}^r)^s$. This signature is valid since

$$e(\sigma(m), g_2) = e(x^s, g_2) \cdot e({g_1}^{rs}, g_2) = e({g_1}^s, y) \cdot e({g_1}^s, {g_2}^r) = e(h(m), y').$$

If the targeted-user forger did not request a signature on a message whose hash is h-dependent, then it eventually fails or outputs an aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, and $k-1$ public keys $(u_2, v_2), \ldots, (u_k, v_k)$, for some positive integer k that is at most n. The co-CDH* solver succeeds if and only if the following events occur:

(E_1) The targeted-user forger does not request a signature on any message whose hash is h-dependent. Since the targeted-user forger makes at most q_s signature queries, this event occurs with probability at least $(1-P)^{q_s}$.

(E_2) The targeted-user forger succeeds after time at most t. Given the first event, the hashing and signing oracles simulated by the solver are indistinguishable from "real" hashing and signing oracles, so $\Pr(E_2 \mid E_1) \geq \epsilon$.

(E_3) The hash of m_1 is h-dependent, while the hash of messages m_2 through m_k are not h-dependent. (If the forged aggregate signature were not on distinct messages, like in the rogue key attack described in Subsection 3.2.1, then this event would not occur and the reduction would fail.) Since the messages are pairwise distinct and k is at most n, this event happens with probability at least $P \cdot (1-P)^{n-1}$.

The solver's success probability ϵ' is therefore

$$\Pr(E_1 \wedge E_2 \wedge E_3) \geq (1-P)^{q_s} \cdot \epsilon \cdot P \cdot (1-P)^{n-1} = \epsilon \cdot P \cdot (1-P)^{q_s+n-1}.$$

By Equation (1.1), the value $P = 1/(n+q_s)$ maximizes this lower bound. Then, applying the approximation for e^{-1} in Equation (1.2) gives the lower bound $\epsilon' \geq \epsilon/(e \cdot (n+q_s))$. If all three events occur, then the valid aggregate signature σ_A satisfies the following equation since it is unique:

$$\sigma_A = h(m_1)^{z+r} \cdot \prod_{i=2}^{k} h(m_i)^{w_i} = (h \cdot {g_1}^{s_1})^{z+r} \cdot \prod_{i=2}^{k} {g_1}^{s_i \cdot w_i} = h^z \cdot h^r \cdot (x \cdot {g_1}^r)^{s_1} \cdot \prod_{i=2}^{k} {u_i}^{s_i}.$$

The solver then computes $h^z = \sigma_A \cdot \left(h^r \cdot (x \cdot {g_1}^r)^{s_1} \cdot \prod_{i=2}^{k} {u_i}^{s_i}\right)^{-1}$.

In total, the co-CDH* solver performs at most $q_h + q_s + n + 3$ exponentiations and $q_h + n + 2$ multiplications in G_1 or G_2. Hence, given a (t, ϵ, q_h, q_s)-targeted-user forger, we can create a $(t', \epsilon', q_h, q_s, m_e)$-co-CDH* solver, where

$$\begin{aligned} t' &= t + (q_h + q_s + n + 3) \cdot T_e + (q_h + n + 2) \cdot T_m, \text{ and} \\ \epsilon' &= \frac{\epsilon}{e \cdot (n + q_s)}. \end{aligned}$$

That is, if the co-CDH* problem is (t', ϵ')-hard, then BGLS is resistant to (t, ϵ, q_h, q_s)-targeted-user forgery, for

$$\begin{aligned} t &= t' - (q_h + q_s + n + 3) \cdot T_e - (q_h + n + 2) \cdot T_m, \text{ and} \\ \epsilon &= \epsilon' \cdot e \cdot (n + q_s). \end{aligned} \qquad \square$$

We believe the type of forgery in the original BGLS security reduction does not reflect the multi-user environments where aggregate signature schemes are deployed—the adversary should not be given a target user to attack. In the following chapter, we examine two new types of adversaries and their effects on tightness of the reduction.

Chapter 4

Improving aggregate signature security definitions

As its name implies, the targeted-user forger is told which user to attack. In a multi-user setting, such an assumption needlessly restricts the adversary. In this chapter, we give two new capabilities to aggregate signature forgers and examine how tightly they correspond to the original forger.

First, we review existing security definitions for multi-signature schemes, where each user signs the same message, and sequential aggregate signature schemes, where the order of aggregation is important. Next, we examine two new types of BGLS forgery: in Section 4.2, we examine forgers that can *expose* users' private keys, and in Section 4.3, forgers that can *replace* users' public keys. In each section, we describe reductions between the new type of forgery and the original targeted-user forgery. Our goal is to argue that targeted-user forgery is harder than our proposed types of forgery. Figure 4.1 on page 24 provides an overview of the tightness gaps of reductions among different types of aggregate signature forgery.

4.1 Comparison to other signature schemes' security models

In this section, we examine assumptions about adversaries in multi-signature schemes and sequential aggregate signature schemes. In identity-based schemes, adversaries often have the power to expose legitimate users' private keys.

Cha and Cheon introduced an attack model for identity-based signatures in 2003 [12]. They require their scheme to be secure against existential forgery under adaptively-chosen message and ID attacks. In this model, the attacker has access to a signing oracle and an extraction oracle that returns the private key corresponding to an ID. In Gentry and Ramzan's identity-based aggregate signature scheme, the adversary can adaptively make key extraction queries—it submits an ID and receives the corresponding signing key [18]. In Bellare and Neven's identity-based multi-signature scheme based on RSA, the adversary has access to a key derivation oracle [6]. Attackers can adaptively corrupt users by

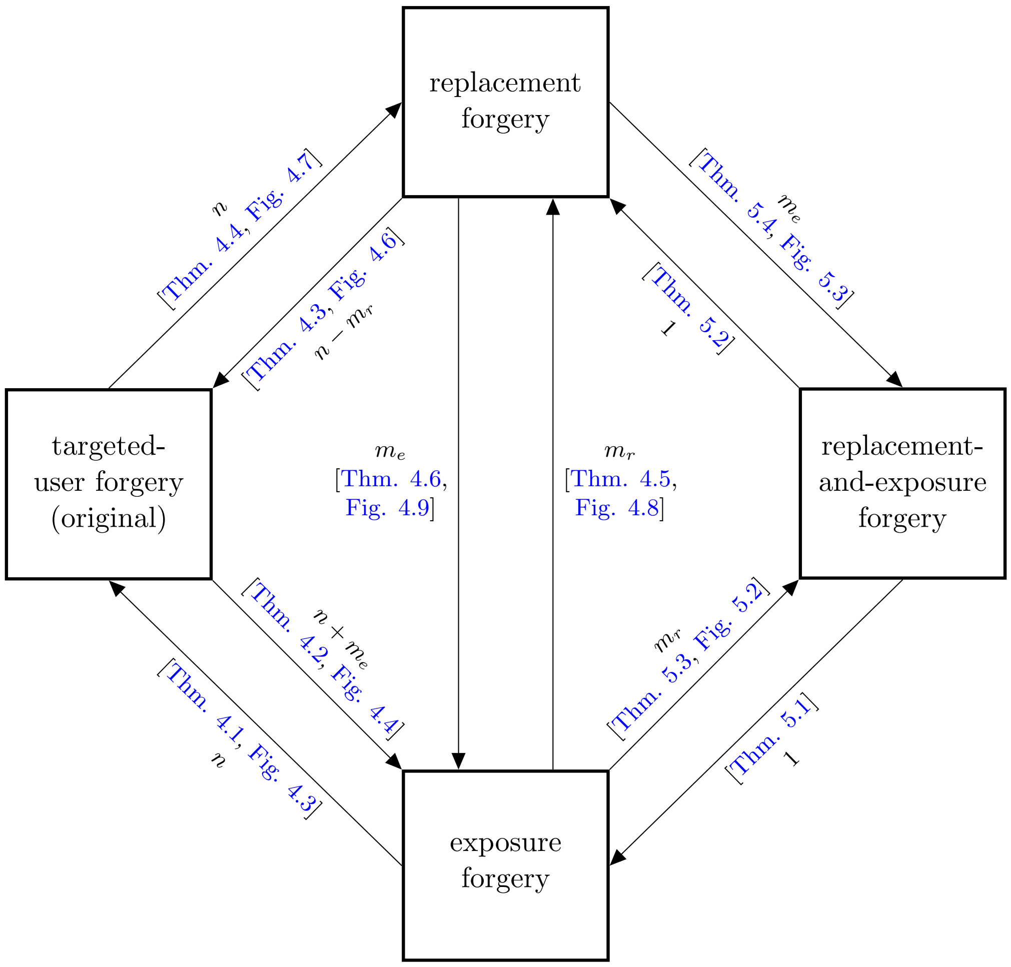

replacement forgery
targeted-user forgery (original)
replacement-and-exposure forgery
exposure forgery
n [Thm. 4.4, Fig. 4.7]
[Thm. 4.3, Fig. 4.6] $n - m_r$
m_e [Thm. 5.4, Fig. 5.3]
[Thm. 5.2] 1
m_e [Thm. 4.6, Fig. 4.9]
m_r [Thm. 4.5, Fig. 4.8]
[Thm. 4.2, Fig. 4.4] $n + m_e$
[Thm. 4.1, Fig. 4.3] n
m_r [Thm. 5.3, Fig. 5.2]
[Thm. 5.1] 1

Figure 4.1: Overview of tightness gaps in reductions between types of aggregate signature forgery for BGLS with type III pairings. The original security reduction for BGLS considers only targeted-user forgers.

submitting their IDs to the key derivation oracle and receiving the corresponding private keys. Although Bellare and Neven admit that some identities may be weaker than others in identity-based schemes, they state that in non-identity-based multi-signature schemes, any honestly-generated public key is as good as any other one. "No adversary is expected to perform significantly better in a model with multiple honest signers, as it could easily have simulated these other signers itself." Bagherzandi and Jarecki's identity-based aggregate and multi-signature schemes based on RSA also allow the attacker to make key derivation queries [2].

In signature schemes that are not identity-based, however, we did not find an example of a security model that allows attackers to learn other users' secret keys. Lysyanskaya, Micali, Reyzin, and Shacham's sequential aggregate signature scheme, based on trapdoor permutations, is secure in the sequential aggregate chosen-key security model, where the adversary receives a single public key [27]. In Lu, Ostrovsky, Sahai, Shacham, and Waters' sequential aggregate signature scheme, the adversary is also given a challenge key [26]. The authors assume that the attacker provides the private key corresponding to the public keys it chooses. Recall from Subsection 3.2.1 that requiring users to prove knowledge of their private keys is one way to prevent the rogue key attack. Bellare, Namprempre, and Neven's unrestricted aggregate signature scheme is not in the chosen-key model either [4]. They highlight the practical advantage of assuming that attackers do not need to know the secret key corresponding to the public keys they choose. The security notion of Neven's sequential aggregate signed data scheme, a generalization of sequential aggregate signatures, also begins with the forger receiving one public key [28].

We argue that attackers could learn other users' private keys not only in identity-based schemes. In any public-key cryptosystem, public keys are exactly that—public. An attacker should not be given one public key to target; it should have its choice from all the public keys it can collect. The next section examines an aggregate forger that can choose which public key to target and, like the forger in an identity-based scheme, can expose other users' private keys.

4.2 Forgers that can expose other users' private keys

In this section, we examine a forger that can choose which users to target and which users to corrupt by exposing their private keys. We denote by m_e the maximum number of private keys the forger can expose. The maximum value of m_e is $n-1$, since a forger automatically fails if it exposes the private keys of all users in the aggregate signature scheme. We introduce this parameter to study its effect on tightness of the reductions.

Definition 4.1.1. An **exposure forger** is an adversary with the following capabilities and goals. It receives n randomly generated public keys $(x_1, y_1), \ldots, (x_n, y_n)$, corresponding to users 1 to n. It adaptively queries a hashing oracle and n individual signing oracles with messages. At any point, the forger can choose to corrupt user j by exposing its private key z_j.
The forger's goal is to output k user indices $\alpha_1, \ldots, \alpha_k$, k distinct messages $m_1, \ldots, m_k$, and a valid aggregate signature σ_A comprising user α_i's signature on message m_i, for each integer i from 1 to k, where k is a positive integer at most n. The forger succeeds if all k user indices correspond to users whose private keys were not exposed, and user α_1 did not provide the forger with a signature on m_1.

Definition 4.1.2. A $(\mathbf{t}, \epsilon, \mathbf{q_h}, \mathbf{q_s}, \mathbf{m_e})$**-exposure forger** runs in time at most t, succeeds with probability at least ϵ, makes at most q_h hashing queries and q_s signing queries, and exposes at most m_e private keys. The success probability is computed over all possible inputs and all of the forger's coin tosses.

Definition 4.1.3. An aggregate signature scheme is **resistant to** $(\mathbf{t}, \epsilon, \mathbf{q_h}, \mathbf{q_s}, \mathbf{m_e})$**-exposure forgery** if no $(t, \epsilon, q_h, q_s, m_e)$-exposure forger exists.

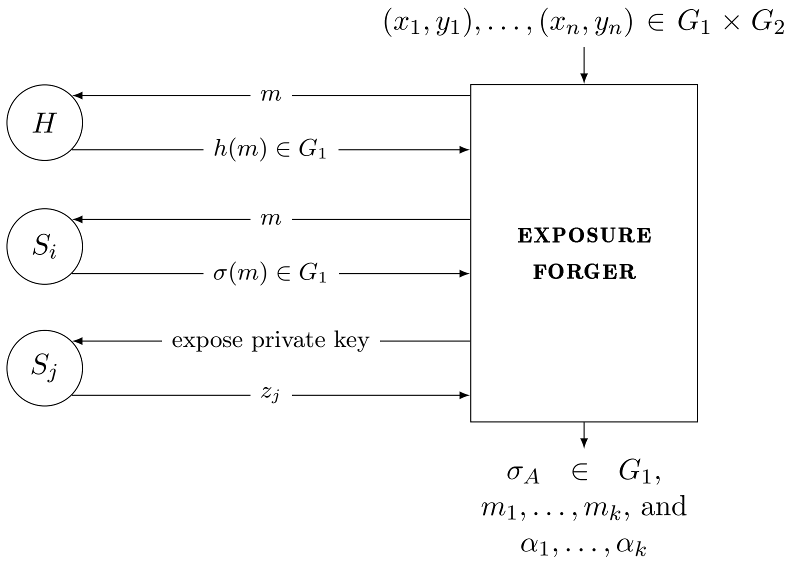

$(x_1, y_1), \ldots, (x_n, y_n) \in G_1 \times G_2$
H
m
$h(m) \in G_1$
S_i
m
$\sigma(m) \in G_1$
S_j
expose private key
z_j
EXPOSURE FORGER
$\sigma_A \in G_1$, $m_1, \ldots, m_k$, and $\alpha_1, \ldots, \alpha_k$

Figure 4.2: Capabilities and goals of an exposure forger.

According to this definition, the exposure forger's aggregate signature must exclude all users whose private keys were exposed. This requirement is without loss of generality for BGLS. The exposure forger can remove the part of the aggregate signature corresponding to user α and message m_α by multiplying the signature with $h(m_\alpha)^{-z_\alpha}$, where z_α is user α's exposed private key. In other aggregate signature schemes—such as those where aggregation is sequential—this assumption may not hold.

Informally, the exposure forger's task seems easier than the targeted-user forger's task because it can choose which user to target. However, it can only determine other users' private keys, not choose them. Which problem is easier? If we relax the forger's goal and prove that the signature scheme is resistant to these forgers, then we are, in theory, making it more secure.

First, we attempt to prove that targeted-user forgery is at least as hard as exposure forgery. Although a reduction does exist, it is not tight. The reduction in the proof of Theorem 4.1 is represented in Figure 4.3 on page 27.

Theorem 4.1 (**exposure forgery $\leq$ targeted-user forgery**)**.** If the BGLS signature scheme in an n-user setting is resistant to $(t', \epsilon', q_h, q_s, m_e)$-exposure forgery, then it is also resistant to (t, ϵ, q_h, q_s)-targeted-user forgery, for

$$t = t' - (q_h + q_s + n - 1) \cdot T_e + (n-1) \cdot T_m, \text{ and}$$
$$\epsilon = \epsilon' \cdot e \cdot n.$$

Proof. We prove the contrapositive of this statement: we show how to build an exposure forger given a targeted-user forger. The exposure forger receives n challenge public keys

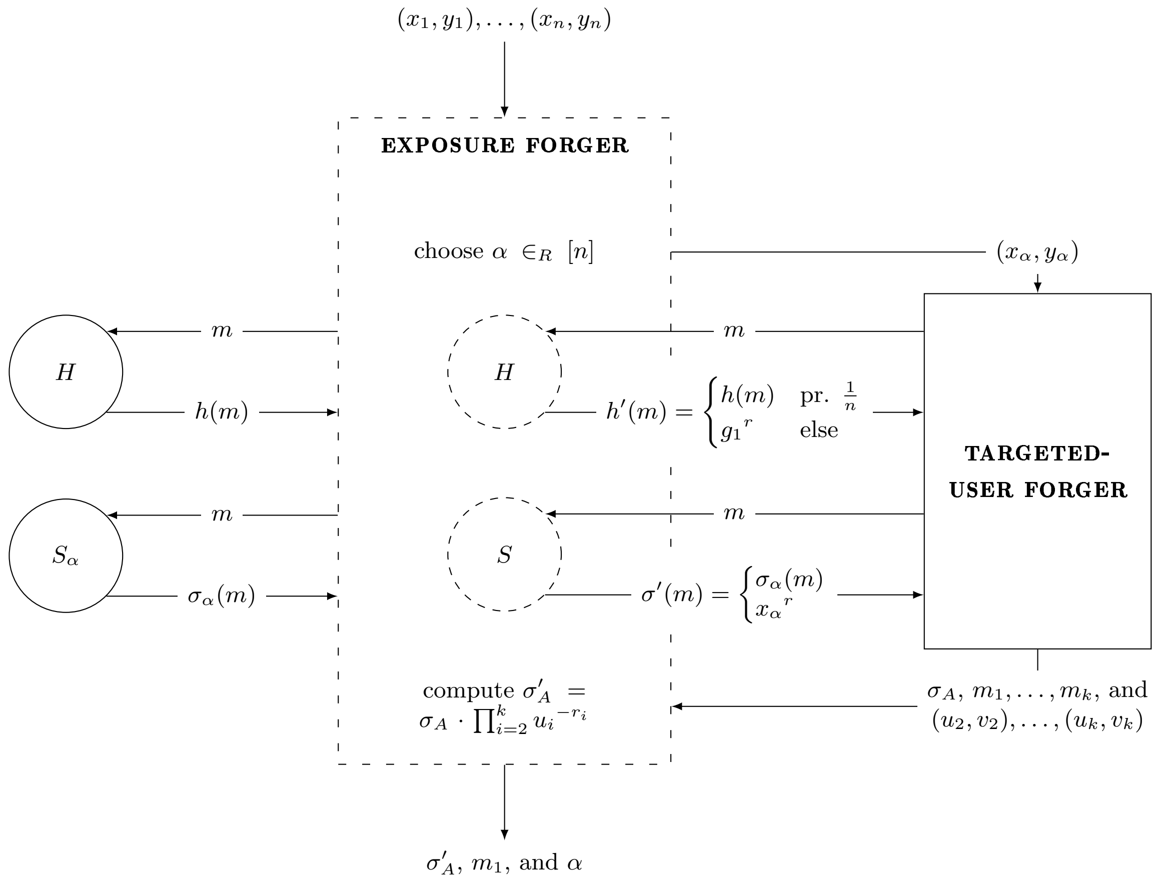

$(x_1, y_1), \ldots, (x_n, y_n)$
EXPOSURE FORGER
choose $\alpha \in_R [n]$
(x_α, y_α)
m
H
$h(m)$
H
$h'(m) = \begin{cases} h(m) & \text{pr. } \frac{1}{n} \\ {g_1}^r & \text{else} \end{cases}$
TARGETED-USER FORGER
m
S_α
$\sigma_\alpha(m)$
S
$\sigma'(m) = \begin{cases} \sigma_\alpha(m) \\ {x_\alpha}^r \end{cases}$
compute $\sigma'_A = \sigma_A \cdot \prod_{i=2}^k {u_i}^{-r_i}$
σ_A, $m_1, \ldots, m_k$, and $(u_2, v_2), \ldots, (u_k, v_k)$
σ'_A, m_1, and α

Figure 4.3: The reduction from exposure forgery to targeted-user forgery has a tightness gap of n, the number of users.

$(x_1, y_1), \ldots, (x_n, y_n)$ and has access to a hashing oracle H and n individual signing oracles S_i. It can expose at most m_e of the users' private keys.

First, the exposure forger must give the targeted-user forger some challenge public key. It randomly chooses one of the public keys it receives, say (x_α, y_α), and passes it to the targeted-user forger. When the targeted-user forger requests the hash of a message, the exposure forger either forwards the query to the hashing oracle, or computes a random power of g_1:

$$h'(m) = \begin{cases} h(m) & \text{with probability } P, \\ {g_1}^r & \text{otherwise.} \end{cases}$$

In the second case, the exposure forger records (m, r), where r is a randomly chosen integer modulo p. We will determine the optimal value of P when computing the exposure forger's success probability.

When the targeted-user forger requests a signature on a message, the exposure forger either forwards the query to user α's signing oracle or computes the appropriate power of x_α:

$$\sigma'(m) = \begin{cases} \sigma_\alpha(m) & \text{if } h'(m) = h(m), \\ {x_\alpha}^r & \text{if } h'(m) = {g_1}^r. \end{cases}$$

The signature in the first case is clearly valid. When the hash is a random power of g_1,

$$e\left(\sigma'(m), g_2\right) = e\left({x_\alpha}^r, g_2\right) = e\left({g_1}^{z_\alpha \cdot r}, g_2\right) = e\left({g_1}^r, y_\alpha\right) = e\left(h'(m), y_\alpha\right),$$

so the signature is also valid in the second case.

Eventually, the targeted-user forger fails or outputs an aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, and $k-1$ public keys $(u_2, v_2), \ldots, (u_k, v_k)$, for some positive integer k that is at most n.

The exposure forger succeeds if and only if the following events occur:

(E_1) The targeted-user forger succeeds after time at most t. From its point of view, the hashing and signing oracles simulated by the exposure forger are indistinguishable from "real" hashing and signing oracles, so $\Pr(E_1) \geq \epsilon$.

(E_2) In the targeted-user forger's output σ_A, the hash of the message signed by the user with the challenge public key is not a random power of g_1, and the hashes of all other messages are random powers of g_1. That is, $h'(m_1) = h(m_1)$ and $h'(m_i) = {g_1}^{r_i}$ for each integer i from 2 to k. Given the targeted-user forger's success, the probability of this event is $\Pr(E_2 \mid E_1) = P \cdot (1-P)^{k-1}$.

The exposure forger's success probability ϵ' is therefore

$$\Pr(E_1 \wedge E_2) = \Pr(E_1) \cdot \Pr(E_2 \mid E_1) \geq \epsilon \cdot P \cdot (1-P)^{k-1} \geq \epsilon \cdot P \cdot (1-P)^{n-1}.$$

By Equation (1.1), the value $P = 1/n$ maximizes this lower bound on the exposure forger's success probability. Then, applying the approximation for e^{-1} in Equation (1.2) gives the lower bound $\epsilon' \geq \epsilon/(e \cdot n)$.

If both events occur, then the exposure forger computes $\sigma'_A = \sigma_A \cdot \prod_{i=2}^{k} {u_i}^{-r_i}$ in G_1. It outputs σ'_A, the user index α, and the message m_1. This aggregate signature is valid since

$$\begin{aligned} e\left(\sigma'_A, g_2\right) &= e\left(\sigma_A, g_2\right) \cdot \prod_{i=2}^{k} e\left({u_i}^{-r_i}, g_2\right) \\ &= \left(e\left(h(m_1), y_\alpha\right) \cdot \prod_{i=2}^{k} e\left({g_1}^{r_i}, v_i\right)\right) \cdot \prod_{i=2}^{k} e\left({u_i}^{-r_i}, g_2\right) \\ &= e\left(h(m_1), y_\alpha\right) \cdot \prod_{i=2}^{k} e\left({g_1}^{r_i}, {g_2}^{w_i}\right) \cdot \prod_{i=2}^{k} e\left({g_1}^{-r_i \cdot w_i}, g_2\right) \\ &= e\left(h(m_1), y_\alpha\right) \cdot \prod_{i=2}^{k} e\left({g_1}^{r_i}, {g_2}^{w_i}\right) \cdot \prod_{i=2}^{k} e\left({g_1}^{-r_i}, {g_2}^{w_i}\right) \\ &= e\left(h(m_1), y_\alpha\right). \end{aligned}$$

The exposure forger does not expose any private keys. Since the targeted-user forger did not request a signature on m_1, the exposure forger did not request a signature on m_1 from user α. It makes at most as many hashing and signing queries as the targeted-user forger, which makes at most q_h and q_s queries. It computes at most $q_h + q_s + n - 1$ exponentiations and $n - 1$ multiplications in G_1 or G_2. Hence, given a (t, ϵ, q_h, q_s)-targeted-user forger, we can create a $(t', \epsilon', q_h, q_s, m_e)$-exposure forger, where

$$\begin{aligned} t' &= t + (q_h + q_s + n - 1) \cdot T_e + (n - 1) \cdot T_m, \text{ and} \\ \epsilon' &= \frac{\epsilon}{e \cdot n}. \end{aligned}$$

That is, if BGLS is resistant to $(t', \epsilon', q_h, q_s, m_e)$-exposure forgery, then it is also resistant to (t, ϵ, q_h, q_s)-targeted-user forgery, for

$$\begin{aligned} t &= t' - (q_h + q_s + n - 1) \cdot T_e - (n - 1) \cdot T_m, \text{ and} \\ \epsilon &= \epsilon' \cdot e \cdot n. \end{aligned}$$

□

In the previous reduction, the exposure forger does not expose any private keys, and we are not aware of a tighter reduction where it does.

Next, we examine the reduction from targeted-user forgery to exposure forgery. Figure 4.4 on page 30 depicts the reduction in the proof of Theorem 4.2.

Theorem 4.2 (targeted-user forgery $\leq$ exposure forgery). If the BGLS signature scheme in an n-user setting is resistant to $(t', \epsilon', q_h, q_s)$-targeted-user forgery, then it is resistant to $(t, \epsilon, q_h, q_s, m_e)$-exposure forgery, for

$$\begin{aligned} t &= t' - (2n + q_s + 1)(T_e + T_m), \text{ and} \\ \epsilon &= \epsilon' \cdot e \cdot (n + m_e). \end{aligned}$$

Proof. We prove the contrapositive of this statement: we show how to build a targeted-user forger given an exposure forger. The targeted-user forger receives one public key (x, y) and has access to a hashing oracle H and signing oracle S.

First, it must give n public keys to the exposure forger. It picks n random integers r_i modulo p and gives it the following n public keys:

$$(x_i, y_i) = \begin{cases} (x \cdot {g_1}^{r_i}, y \cdot {g_2}^{r_i}) & \text{with probability } P, \\ ({g_1}^{r_i}, {g_2}^{r_i}) & \text{otherwise.} \end{cases}$$

The targeted-user forger records each integer i and random integer r_i modulo p. We will determine the optimal value of P when analyzing the targeted-user forger's success probability.

When the exposure forger requests the hash of a message m, the targeted-user forger simply forwards the message to the hashing oracle and returns the unchanged result $h(m)$.

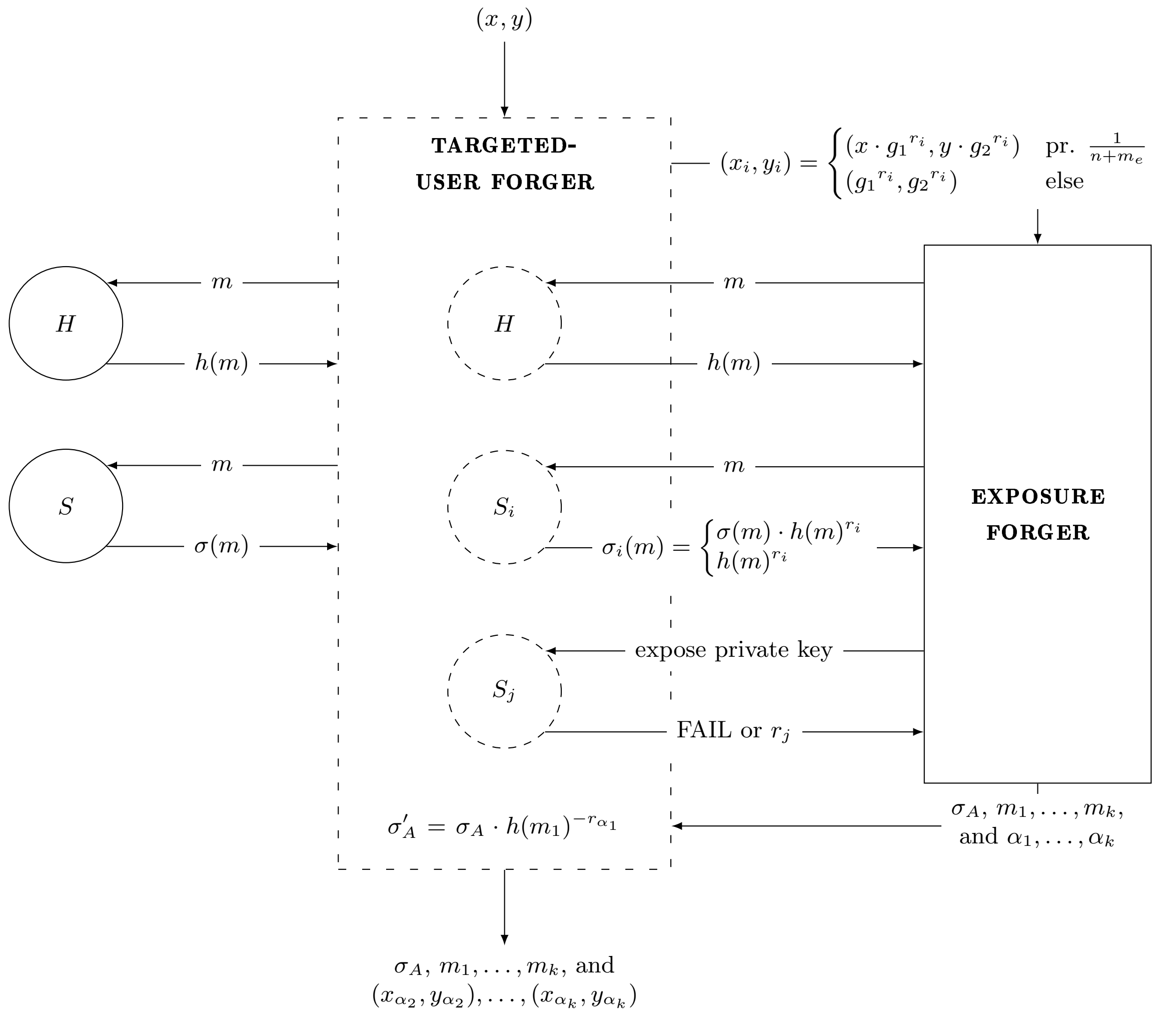

(x, y)
TARGETED-USER FORGER
$(x_i, y_i) = \begin{cases} (x \cdot {g_1}^{r_i}, y \cdot {g_2}^{r_i}) & \text{pr. } \frac{1}{n+m_e} \\ ({g_1}^{r_i}, {g_2}^{r_i}) & \text{else} \end{cases}$
H
m
$h(m)$
S
$\sigma(m)$
S_i
$\sigma_i(m) = \begin{cases} \sigma(m) \cdot h(m)^{r_i} \\ h(m)^{r_i} \end{cases}$
S_j
expose private key
FAIL or r_j
EXPOSURE FORGER
$\sigma'_A = \sigma_A \cdot h(m_1)^{-r_{\alpha_1}}$
σ_A, $m_1, \ldots, m_k$, and $\alpha_1, \ldots, \alpha_k$
σ_A, $m_1, \ldots, m_k$, and $(x_{\alpha_2}, y_{\alpha_2}), \ldots, (x_{\alpha_k}, y_{\alpha_k})$

Figure 4.4: The reduction from targeted-user forgery to exposure forgery has a tightness gap of $n + m_e$.

When the exposure forger requests a signature on a message m by user i, the targeted-user forger first looks up the integer r_i. If user i's public key is (x, y)-dependent, then the targeted-user forger forwards m to the real signing oracle to get $\sigma(m)$, a signature on m by the user with public key (x, y). It then multiplies $\sigma(m)$ by $h(m)^{r_i}$ and returns this product to the exposure forger. If user i's public key is not (x, y)-dependent, then the targeted-user forger simply computes $h(m)^{r_i}$ and returns this value to the exposure forger as the user i's signature on m:

$$\sigma'_i(m) = \begin{cases} \sigma(m) \cdot h(m)^{r_i} & \text{if user } i\text{'s public key is } (x, y)\text{-dependent,} \\ h(m)^{r_i} & \text{otherwise.} \end{cases}$$

These signatures are both valid. First, when user i's public key is (x, y)-dependent,

$$e\left(\sigma'_i(m), g_2\right) = e\left(\sigma(m), g_2\right) \cdot e\left(h(m)^{r_i}, g_2\right) = e\left(h(m), y\right) \cdot e\left(h(m), {g_2}^{r_i}\right) = e\left(h(m), y_i\right).$$

Second, when user i's public key is not (x, y)-dependent,

$$e\left(\sigma_i'(m), g_2\right) = e\left(h(m)^{r_i}, g_2\right) = e\left(h(m), y_i\right).$$

At any point, the exposure forger may request user j's private key. If user j's public key is (x, y)-dependent, then the targeted-user forger fails since it cannot answer. However, if user j's public key is not (x, y)-dependent, then the targeted-user forger looks up the integer r_j and returns this value to the exposure forger.

If the targeted-user forger can answer each of the exposure forger's exposure queries, then eventually the exposure forger fails or outputs a valid aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, and k user indices $\alpha_1, \ldots, \alpha_k$, for some positive integer k that is at most n. The forged aggregate signature satisfies $e\left(\sigma_A, g_2\right) = \prod_{i=1}^{k} e\left(h(m_i), y_{\alpha_i}\right)$ and the exposure forger did not expose the private key of any user α_i, nor did it request a signature on m_1 from user α_1.

The targeted-user forger can construct a forgery if and only if the four following events occur:

(E_1) The targeted-user forger can answer each of the exposure forger's queries to obtain users' private keys. That is, each user whose private key is requested has an (x, y)-independent public key. Since the exposure forger can expose at most m_e private keys, this event happens with probability at least $(1 - P)^{m_e}$.

(E_2) The exposure forger succeeds after time at most t. Given the first event, this event happens with probability at least ϵ, since the hashing oracle and signing oracle simulated by the targeted-user forger are indistinguishable from a random oracle and a real signing oracle.

(E_3) User α_1 has an (x, y)-dependent public key. Given the first two events, this event happens with probability at least P.

(E_4) The exposure forger does not request signatures on message m_1 by any user whose public key is (x, y)-dependent. That is, every time the exposure forger requests a signature on m_1, it is from a user whose public key is not (x, y)-dependent. This event is independent from the others and happens with probability at least $(1 - P)^{n-1}$.

When all four of these events occur, the targeted-user forger computes the signature $\sigma_A' = \sigma_A \cdot h(m_1)^{-r_{\alpha_1}}$. It outputs σ_A', the messages $m_1, \ldots, m_k$, and the $k - 1$ public keys $(x_2, y_2), \ldots, (x_k, y_k)$.

This signature is valid since

$$\begin{aligned}
e\left(\sigma'_A, g_2\right) &= e\left(\sigma_A, g_2\right) \cdot e\left(h(m_1)^{-r_{\alpha_1}}, g_2\right) \\
&= \prod_{i=1}^{k} e\left(h(m_i), y_{\alpha_i}\right) \cdot e\left(h(m_1)^{-r_{\alpha_1}}, g_2\right) \\
&= e\left(h(m_1), y\right) \cdot e\left(h(m_1), {g_2}^{r_{\alpha_1}}\right) \cdot e\left(h(m_1), {g_2}^{-r_{\alpha_1}}\right) \cdot \prod_{i=2}^{k} e\left(h(m_i), y_{\alpha_i}\right) \\
&= e\left(h(m_1), y\right) \cdot \prod_{i=2}^{k} e\left(h(m_i), y_{\alpha_i}\right).
\end{aligned}$$

The targeted-user forger's success probability ϵ' has the following lower bound:

$$\epsilon' \geq \Pr\left(E_4\right) \cdot \Pr\left(E_3 \mid E_2 \wedge E_1\right) \cdot \Pr\left(E_2 \mid E_1\right) \cdot \Pr\left(E_1\right) \geq \epsilon \cdot P \cdot (1-P)^{n+m_e-1}.$$

By Equation (1.1), the value $P = 1/(n+m_e)$ maximizes this lower bound. Then, using the approximation for e^{-1} (Equation (1.2)), we conclude that the targeted-user forger's success probability ϵ' is at least $\epsilon/(e \cdot (n+m_e))$.

The targeted-user forger computes at most $2n + q_s + 1$ exponentiations and $2n + q_s + 1$ multiplications in G_1 or G_2. It makes the same number of hashing queries and at most the same number of signing queries as the exposure forger. Hence, we can build a $(t', \epsilon', q_h, q_s)$-targeted-user forger from a $(t, \epsilon, q_h, q_s, m_e)$-exposure forger, where

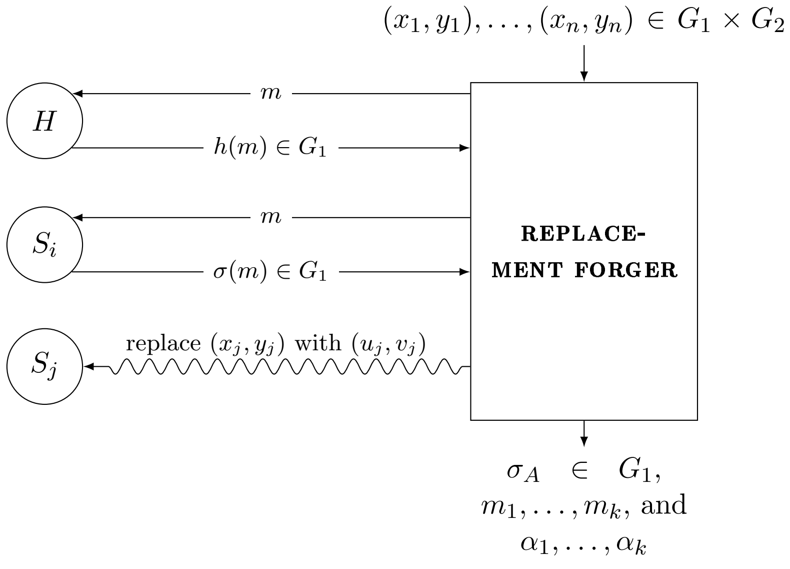
$$\begin{aligned}
t' &= t + (2n + q_s + 1)(T_e + T_m), \text{ and} \\
\epsilon' &= \frac{\epsilon}{e \cdot (n + m_e)}.
\end{aligned}$$

□

The reductions between targeted-user forgery and exposure forgery lose tightness by the number of users, n, in both directions. It is unclear which problem is harder. In the next section, we examine a forger that can replace users' public keys instead of just exposing their private keys.

4.3 Forgers that can replace other users' public keys

The replacement forger can replace a user's public key with any public key it chooses. The parameter m_r is an upper bound on the number public keys it can replace. The strongest replacement forger could replace $n-1$ public keys, but we introduce this parameter to study its effect on tightness. Since our reductions are in the plain public-key model, we cannot assume that the replacement forger knows the private key corresponding to the public key it chooses. Hence, we allow replacement forgers to include messages signed by users whose public keys it replaced.



$(x_1, y_1), \ldots, (x_n, y_n) \in G_1 \times G_2$
H
m
$h(m) \in G_1$
S_i
m
$\sigma(m) \in G_1$
REPLACE-MENT FORGER
S_j
replace (x_j, y_j) with (u_j, v_j)
$\sigma_A \in G_1$, $m_1, \ldots, m_k$, and $\alpha_1, \ldots, \alpha_k$

Figure 4.5: Capabilities and goals of a replacement forger.

Definition 4.2.1. A **replacement forger** is an adversary with the following capabilities and goals. It receives n randomly generated public keys $(x_1, y_1), \ldots, (x_n, y_n)$, corresponding to users 1 to n. It adaptively queries a hashing oracle and n individual signing oracles with messages. At any point, the forger can choose to corrupt a user by replacing its public key with any (u, v) of its choice in $G_1 \times G_2$. (Of course, the forger cannot then request signatures from that user.)
The forger's goal is to output k user indices $\alpha_1, \ldots, \alpha_k$, k distinct messages $m_1, \ldots, m_k$, and a valid aggregate signature σ_A comprising user α_i's signature on message m_i for each i from 1 to k, where k is a positive integer at most n. The aggregate signature will satisfy $e(\sigma_A, g_2) = \prod_{i=1}^{k} e\left(h(m_i), y'_{\alpha_i}\right)$ where

$$y'_{\alpha_i} = \begin{cases} v_{\alpha_i} & \text{if the forger replaced user } \alpha_i\text{'s public key } (x_{\alpha_i}, y_{\alpha_i}) \text{ with } (u_{\alpha_i}, v_{\alpha_i}), \\ y_{\alpha_i} & \text{otherwise.} \end{cases}$$

The forger succeeds if it did not replace user α_1's public key, if user α_1 did not give the forger a signature on m_1, and if the replacement forger made no more than m_r public key replacements.

Definition 4.2.2. A $(\mathbf{t}, \epsilon, \mathbf{q_h}, \mathbf{q_s}, \mathbf{m_r})$**-replacement forger** makes at most q_h hashing queries and q_s signing queries, replaces at most m_r public keys, runs in time at most t, and succeeds with probability at least ϵ. The success probability is computed over all possible inputs $(x_1, y_1), \ldots, (x_n, y_n)$ and all of the forger's coin tosses.

Definition 4.2.3. An aggregate signature scheme is **resistant to** $(\mathbf{t}, \epsilon, \mathbf{q_h}, \mathbf{q_s}, \mathbf{m_r})$**-replacement forgery** if no $(t, \epsilon, q_h, q_s, m_r)$-replacement forger exists.

How does this type of forgery compare to the problem of targeted-user forgery? If replacement forgery is easier, then we obtain a stronger security definition for aggregate signature schemes. In Theorem 4.3, we examine a reduction from the problem of replacement forgery to the problem of targeted-user forgery. When we assume that replacement forgers can replace all but one public key, this reduction is tight: the problem of replacement forgery is no harder than the problem of targeted-user forgery. The reduction in the proof of this theorem is depicted in Figure 4.6 on page 35.

Theorem 4.3 (replacement forgery $\leq$ targeted-user forgery). If the BGLS aggregate signature scheme in an n-user setting is resistant to $(t', \epsilon', q_h, q_s, m_r)$-replacement forgery, then it is also resistant to (t, ϵ, q_h, q_s)-targeted-user forgery, for

$$\begin{aligned} t &= t' - (n-1) \cdot T_m - (q_h + q_s + n - 1) \cdot T_e, \text{ and} \\ \epsilon &= \epsilon' \cdot e \cdot (n - m_r). \end{aligned}$$

Proof. We prove the contrapositive of this statement: we show how to build a replacement forger given a targeted-user forger. The replacement forger receives n public keys $(x_1, y_1), \ldots, (x_n, y_n)$, can query a hashing oracle H, and can query a signing oracle S_i for each user i. It randomly picks an integer α_1 between 1 and n and gives the targeted-user forger the public key $(x_{\alpha_1}, y_{\alpha_1})$.

When the targeted-user forger requests the hash of a message, the replacement forger either forwards the request to the hashing oracle H or computes a random power of g_1:

$$h'(m) = \begin{cases} h(m) & \text{with probability } P, \\ {g_1}^r & \text{otherwise.} \end{cases}$$

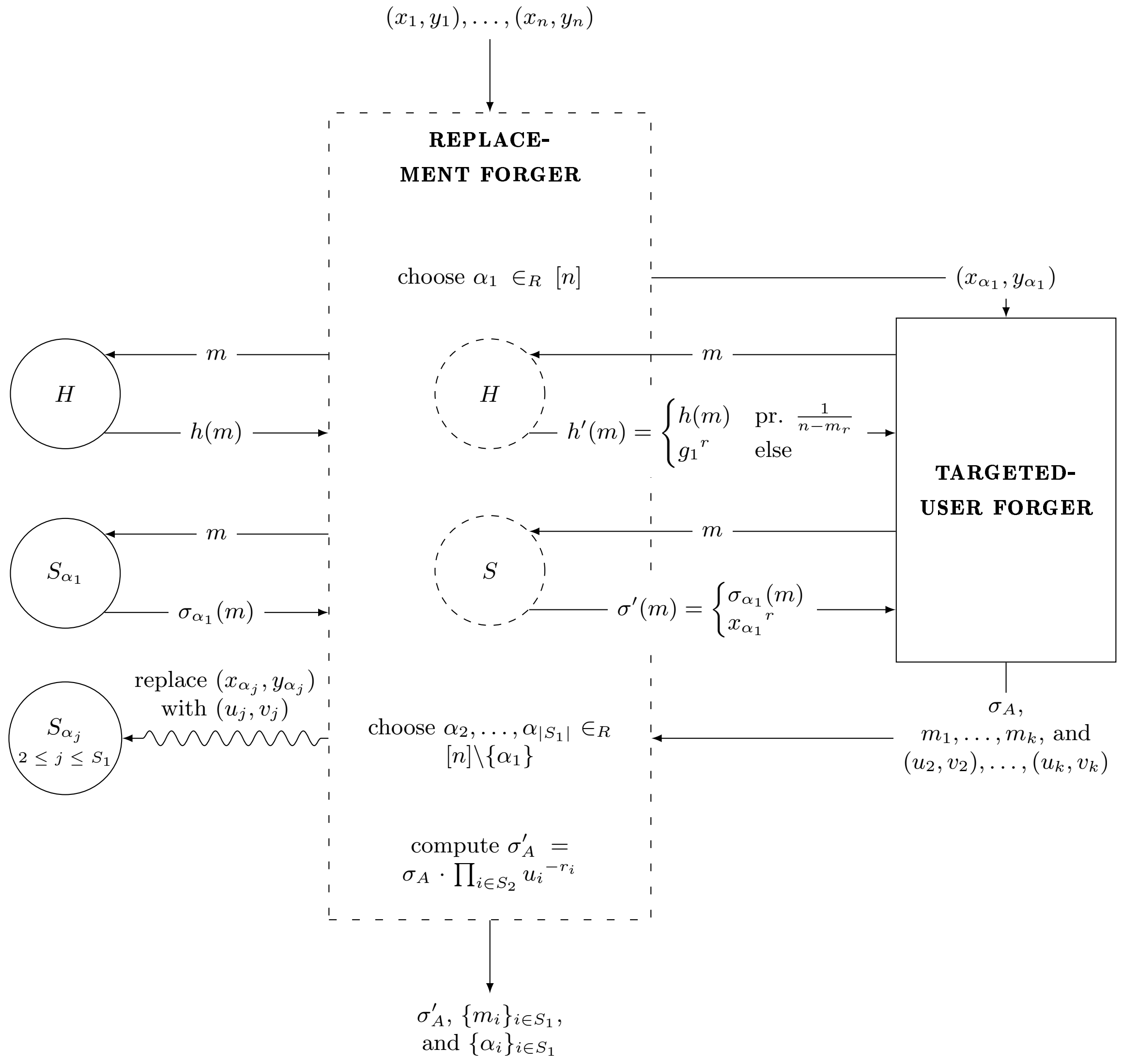
The replacement forger records the message m and the random integer r modulo p. We will determine the optimal value of the probability P when we compute the replacement forger's success probability.

When the targeted-user forger requests a signature, the replacement forger must either request a signature on the same message from the signing oracle S_{α_1} or compute the appropriate power of x_{α_1}:

$$\sigma'(m) = \begin{cases} \sigma_{\alpha_1}(m) & \text{if } h'(m) = h(m), \\ {x_{\alpha_1}}^r & \text{if } h'(m) = {g_1}^r. \end{cases}$$

After time at most t and with some probability, the targeted-user forger outputs a valid forged aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, and $k-1$ public keys $(u_2, v_2), \ldots, (u_k, v_k)$ of its choice, for some positive integer k that is at most n. Partition the integers from 1 to k into the following two sets:

$$\begin{aligned} S_1 &= \{i \in [k] \mid h'(m_i) = h(m_i)\} \text{ and} \\ S_2 &= \{i \in [k] \mid h'(m_i) = {g_1}^{r_i}\}. \end{aligned}$$



$(x_1, y_1), \ldots, (x_n, y_n)$
REPLACE-MENT FORGER
choose $\alpha_1 \in_R [n]$
$(x_{\alpha_1}, y_{\alpha_1})$
m
H
$h(m)$
m
H
$h'(m) = \begin{cases} h(m) & \text{pr. } \frac{1}{n-m_r} \\ {g_1}^r & \text{else} \end{cases}$
TARGETED-USER FORGER
m
S_{α_1}
$\sigma_{\alpha_1}(m)$
m
S
$\sigma'(m) = \begin{cases} \sigma_{\alpha_1}(m) \\ {x_{\alpha_1}}^r \end{cases}$
replace $(x_{\alpha_j}, y_{\alpha_j})$ with (u_j, v_j)
S_{α_j} $2 \leq j \leq S_1$
choose $\alpha_2, \ldots, \alpha_{|S_1|} \in_R [n] \backslash \{\alpha_1\}$
σ_A, $m_1, \ldots, m_k$, and $(u_2, v_2), \ldots, (u_k, v_k)$
compute $\sigma'_A = \sigma_A \cdot \prod_{i \in S_2} {u_i}^{-r_i}$
σ'_A, $\{m_i\}_{i \in S_1}$, and $\{\alpha_i\}_{i \in S_1}$

Figure 4.6: The reduction from replacement forgery to targeted-user forgery has a tightness gap of $n - m_r$, so it is tight when $m_r = n - 1$.

The replacement forger succeeds if and only if the following events occur:

(E_1) The targeted-user forger succeeds after time at most t. Since the hashing and signing oracles simulated by the replacement forger are indistinguishable from "real" oracles, this event happens with probability at least ϵ.

(E_2) The index 1 is in S_1; m_1 has a real hash. The probability of this event given the targeted-user forger's success is P.

(E_3) At most m_r indices other than 1 are in S_1; at least $k-1-m_r$ indices are in S_2. Given the first two events, this event happens with probability

$$\sum_{i=k-1-m_r}^{k-1} \Pr\left(\text{exactly } i \text{ indices are in } S_2\right) = \sum_{i=k-1-m_r}^{k-1} \binom{k-1}{i}(1-P)^i P^{k-1-i},$$

but the probability that any $k-1-m_r$ (or more) indices are in S_2 is at least the probability that some particular $k-1-m_r$ indices are in S_2. Therefore, $\Pr\left(E_3 \mid E_1 \wedge E_2\right) \geq (1-P)^{k-1-m_r}$.

When these three required events occur, the replacement forger computes $\sigma'_A = \sigma_A \cdot \prod_{i \in S_2} {u_i}^{-r_i}$. Suppose, without loss of generality, that the indices in S_1 are $1, \ldots, |S_1|$. The solver chooses $|S_1|-1$ random, distinct indices $\alpha_2, \ldots, \alpha_{|S_1|}$ between 1 and n that are different from α_1. For each index j in $S_1 \setminus \{1\} = \{2, \ldots, |S_1|\}$, it replaces the public key of user α_j with $(u'_{\alpha_j}, v'_{\alpha_j}) = (u_j, v_j)$. Finally, the replacement forger outputs the aggregate signature σ'_A, the messages m_i for each $i \in S_1$, and the user indices α_i for $i \in S_1$. This signature is valid since

$$\begin{aligned} e(\sigma'_A, g_2) &= e(\sigma_A, g_2) \cdot \prod_{i \in S_2} e({u_i}^{-r_i}, g_2) \\ &= e(h(m_1), y_{\alpha_1}) \cdot \prod_{i \in S_1 \setminus \{1\}} e(h(m_i), v_i) \cdot \prod_{i \in S_2} e({g_1}^{r_i}, v_i) \cdot \prod_{i \in S_2} e({g_1}^{-r_i}, v_i) \\ &= e(h(m_1), y_{\alpha_1}) \cdot \prod_{i \in S_1 \setminus \{1\}} e(h(m_i), v'_{\alpha_i}). \end{aligned}$$

The three events are necessary and sufficient for the replacement forger to succeed. Since the positive integer k is at most n, the replacement forger's success probability ϵ' is at least $\epsilon \cdot P \cdot (1-P)^{n-1-m_r}$. By Equation (1.1), the value $P = 1/(n-m_r)$ maximizes the lower bound. We apply Equation (1.2) to obtain the following lower bound on the targeted-user forger's success probability ϵ':

$$\epsilon' \geq \frac{\epsilon}{e \cdot (n-m_r)}.$$

The replacement forger computes at most $|S_2|$ multiplications and $q_h + q_s + |S_2|$ exponentiations. It makes at most as many hashing queries and signing queries as the targeted-user forger, and it replaces at most m_r public keys. Hence, given a (t, ϵ, q_h, q_s)-targeted-user forger, it is possible to build a $(t', \epsilon', q_h, q_s, m_r)$-replacement forger, for

$$t' = t + (n-1) \cdot T_m + (q_h + q_s + n - 1) \cdot T_e, \text{ and}$$
$$\epsilon' = \frac{\epsilon}{e \cdot (n - m_r)}. \qquad \square$$

When $m_r = n - 1$, the reduction is tight: for the most powerful replacement forgers, the problem of replacement forgery is no harder than targeted-user forgery.

Next, we devise a reduction from targeted-user forgery to replacement forgery. We describe the proof in Theorem 4.4 and depict the reduction in Figure 4.7 on page 38.

Theorem 4.4 (targeted-user forgery $\leq$ replacement forgery). If the BGLS signature scheme in an n-user setting is resistant to $(t', \epsilon', q_h, q_s)$-targeted-user forgery, then it is resistant to $(t, \epsilon, q_h, q_s, m_r)$-replacement forgery, for

$$t = t' - (2n + q_s + 1)(T_m + T_e), \text{ and}$$
$$\epsilon = \epsilon' \cdot e \cdot n.$$

Proof. We prove the contrapositive of this statement: we show how to build a targeted-user forger given a replacement forger. The targeted-user forger receives a public key (x, y) and has access to a hashing oracle H and a signing oracle S. For each integer i from 1 to n, the targeted-user forger picks a random integer r_i modulo p and gives the following public key to the replacement forger:

$$(x_i, y_i) = \begin{cases} (x \cdot g^{r_i}, y \cdot g^{r_i}) & \text{with probability } P, \\ ({g_1}^{r_i}, {g_2}^{r_i}) & \text{otherwise.} \end{cases}$$

For each of the n public keys, the targeted-user forger records (i, r_i) and whether the key is (x, y)-dependent. We will determine the optimal value of the probability P when analyzing the targeted-user forger's success probability.

Whenever the replacement forger requests the hash of a message m, the targeted-user forger simply passes on the request to the hashing oracle H and directly returns the result, $h(m)$. Whenever the replacement forger requests a signature by user i on a message m, the targeted-user forger computes $h(m)^{r_i}$. If user i's public key is not (x, y)-dependent, then the targeted-user forger gives this signature to the replacement forger. However, if its public key is (x, y)-dependent, then it forwards the signature request to the signing oracle S. It multiplies the result, $\sigma(m)$, with $h(m)^{r_i}$:

$$\sigma'_i(m) = \begin{cases} h(m)^{r_i} & \text{if user } i\text{'s public key is not } (x, y)\text{-dependent,} \\ \sigma_i(m) \cdot h(m)^{r_i} & \text{if user } i\text{'s public key is } (x, y)\text{-dependent.} \end{cases}$$

The replacement forger can, at any time, replace user j's public key with any (u_j, v_j) in $G_1 \times G_2$. The targeted-user forger then records j and (u_j, v_j).

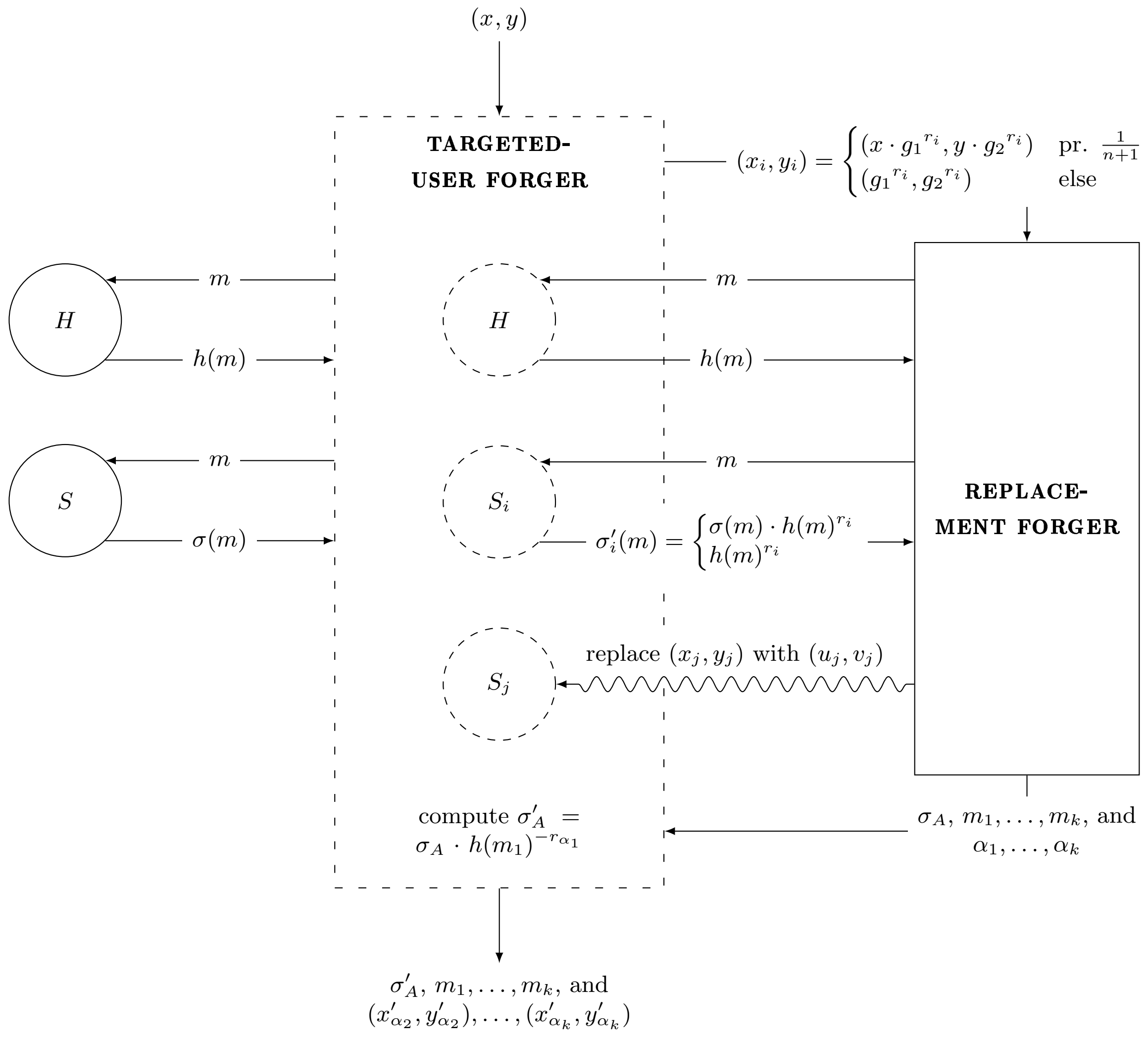

(x, y)
TARGETED-USER FORGER
$(x_i, y_i) = \begin{cases} (x \cdot g_1{}^{r_i}, y \cdot g_2{}^{r_i}) & \text{pr. } \frac{1}{n+1} \\ (g_1{}^{r_i}, g_2{}^{r_i}) & \text{else} \end{cases}$
m
H
$h(m)$
S
$\sigma(m)$
S_i
$\sigma_i'(m) = \begin{cases} \sigma(m) \cdot h(m)^{r_i} \\ h(m)^{r_i} \end{cases}$
REPLACEMENT FORGER
S_j
replace (x_j, y_j) with (u_j, v_j)
compute $\sigma_A' = \sigma_A \cdot h(m_1)^{-r_{\alpha_1}}$
σ_A, $m_1, \ldots, m_k$, and $\alpha_1, \ldots, \alpha_k$
σ_A', $m_1, \ldots, m_k$, and $(x'_{\alpha_2}, y'_{\alpha_2}), \ldots, (x'_{\alpha_k}, y'_{\alpha_k})$

Figure 4.7: The reduction from targeted-user forgery to replacement forgery has a tightness gap of n.

After time at most t, the replacement forger either fails or outputs a valid forged aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, and k user indices $\alpha_1, \ldots, \alpha_k$ for some positive integer k that is at most n. Let $(x'_1, y'_1), \ldots, (x'_n, y'_n)$ denote the public keys after the replacement forger terminates:

$$(x'_i, y'_i) = \begin{cases} (u_i, v_i) & \text{if the replacement forger replaced } (x_i, y_i), \\ (x_i, y_i) & \text{otherwise.} \end{cases}$$

The valid forged signature satisfies $e(\sigma_A, g) = \prod_{i=1}^{k} e(h(m_i), y'_{\alpha_i})$ and the replacement forger did not replace user α_1's public key, nor did it request a signature from this user on m_1. The targeted-user forger succeeds if and only if the following three events happen:

(E_1) The replacement forger succeeds after time at most t. Since the hashing and signing oracle simulated by the targeted-user forger are indistinguishable from a random oracle and a "real" signing oracle, $\Pr(E_1) \geq \epsilon$.

(E_2) User α_1's public key is (x, y)-dependent. Given the replacement forger's success, this event happens with probability $\Pr(E_2 \mid E_1) = P$.

(E_3) The replacement forger did not request a signature on message m_1 from any user whose public key is (x, y)-dependent. Given the first two events, this event happens with probability at least $(1-P)^{n-1}$ since there are at most $n-1$ users from whom the replacement forger could request signatures on m_1.

The probability of these three events occurring is at least $\epsilon \cdot P \cdot (1-P)^{n-1}$. By Equation (1.1), the value $P = 1/n$ maximizes this lower bound. We apply Equation (1.2) to obtain the following lower bound on the targeted-user forger's success probability ϵ':

$$\epsilon' \geq \frac{\epsilon}{e \cdot n}.$$

When all three events occur, the targeted-user forger computes $\sigma'_A = \sigma_A \cdot h(m_1)^{-r_{\alpha_1}}$. It outputs the aggregate signature σ'_A, the k distinct messages $m_1, \ldots, m_k$, and the $k-1$ public keys $(x'_{\alpha_2}, y'_{\alpha_2}), \ldots, (x'_{\alpha_k}, y'_{\alpha_k})$. The targeted-user forger's aggregate signature σ'_A is valid since

$$\begin{aligned} e(\sigma'_A, g) &= e(\sigma_A, g_2) \cdot e\left(h(m_1)^{-r_{\alpha_1}}, g_2\right) \\ &= \prod_{i=1}^{k} e\left(h(m_i), y'_{\alpha_i}\right) \cdot e\left(h(m_1), {g_2}^{-r_{\alpha_1}}\right) \\ &= e\left(h(m_1), y \cdot {g_2}^{r_{\alpha_1}}\right) \cdot e\left(h(m_2), y'_{\alpha_2}\right) \cdots e\left(h(m_k), y'_{\alpha_k}\right) \cdot e\left(h(m_1), {g_2}^{-r_{\alpha_1}}\right) \\ &= e\left(h(m_1), y\right) \cdot e\left(h(m_2), y'_{\alpha_2}\right) \cdots e\left(h(m_k), y'_{\alpha_k}\right). \end{aligned}$$

The time t' required for the targeted-user forger to output a forgery is at most $t + (2n + q_s + 1)(T_m + T_e)$. Hence, given a $(t, \epsilon, q_h, q_s, m_r)$-replacement forger, we can build a $(t', \epsilon', q_h, q_s)$-targeted-user forger, for

$$\begin{aligned} t' &= t + (2n + q_s + 1)(T_m + T_e), \text{ and} \\ \epsilon' &= \frac{\epsilon}{e \cdot n}. \end{aligned}$$

□

If the number of signing queries q_s the replacement forger can make is less than the number of users n, then this reduction loses tightness by q_s instead of n. When computing the probability of E_3's occurrence, the number of signature queries the replacement forger can make on m_1 is bounded above by q_s. The optimal value of P is $1/q_s$ in this case. Regardless of the relation between n and q_s or the value of m_r, we do not know of a tight reduction from targeted-user forgery to replacement forgery. However, the problem of replacement forgery does tightly reduce to the problem of targeted-user forgery when the replacement forger can replace all but one public key. In this case, the problem of replacement forger is no harder than the problem of targeted-user forger.

4.4 Is exposure forgery or replacement forgery easier?

In Section 4.3, we proved that replacement forgery is no harder than targeted-user forgery. The implications of our reductions between exposure forgery and targeted-user forgery in Section 4.2 were unclear since non-tight reductions exist in both directions. How does the problem of exposure forgery compare to the problem of replacement forgery? Theorem 4.5 and Figure 4.8 on page 41 examine a reduction from exposure forgery to replacement forgery.

Theorem 4.5 (exposure forgery $\leq$ replacement forgery). If the BGLS signature scheme in an n-user setting is resistant to $(t', \epsilon', q_h, q_s, m_e)$-exposure forgery, then it is resistant to $(t, \epsilon, q_h, q_s, m_r)$-replacement forgery, for

$$\begin{aligned} t &= t' - (q_h + q_s + n - 1) \cdot T_e - (n-1) \cdot T_m, \\ \epsilon &= \epsilon' \cdot e \cdot (m_r + 1), \end{aligned}$$

and any positive integer m_r that is at most $n-1$.

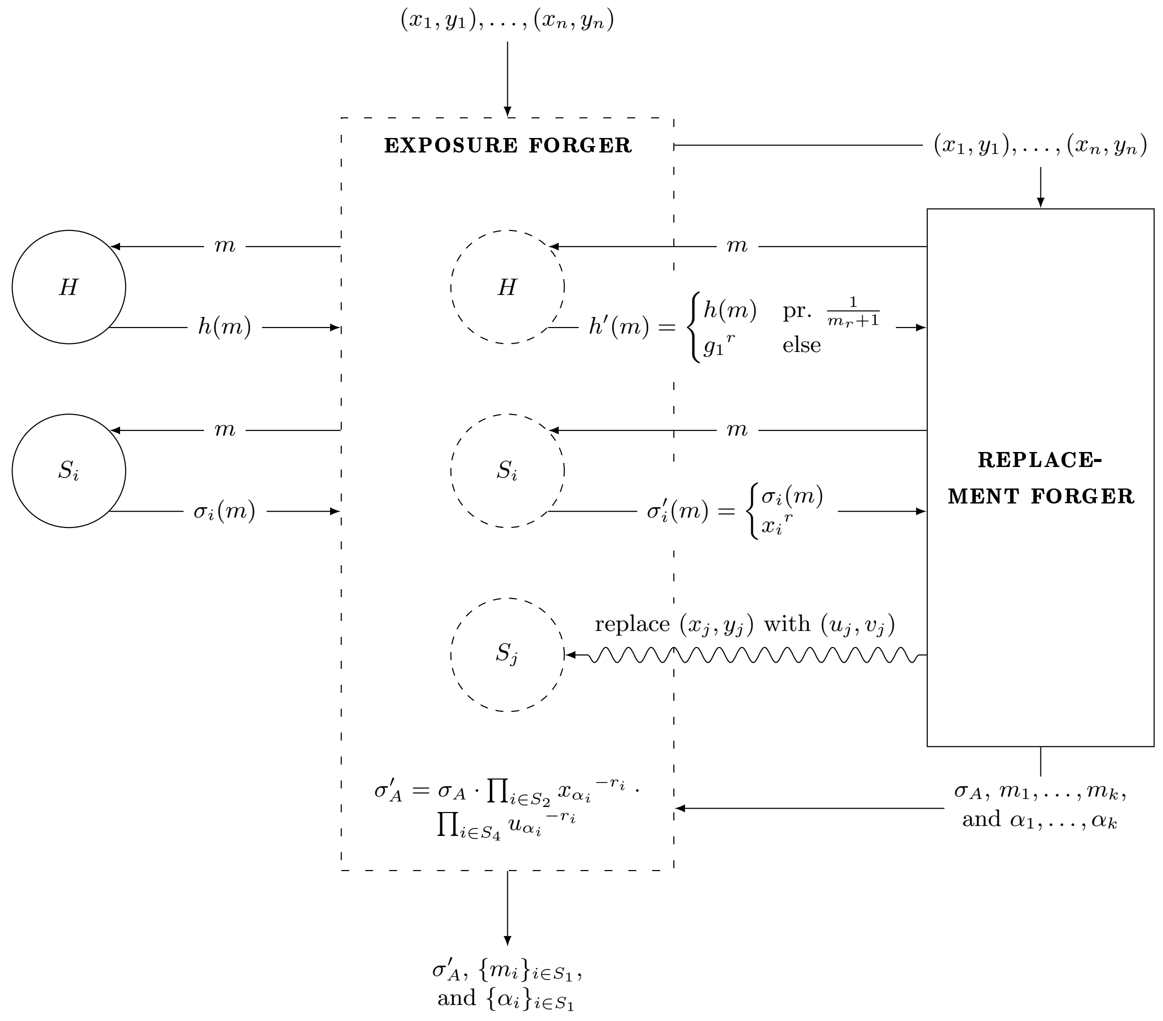
Proof. We prove the contrapositive of this statement: we describe how to build an exposure forger given a replacement forger. The exposure forger receives n public keys $(x_1, y_1), \ldots, (x_n, y_n)$ and has access to a hashing oracle H and n individual signing oracles S_i. First, it gives the same n public keys to the replacement forger.

When the replacement forger requests the hash of a message m, the exposure forger either forwards the hashing query to H and returns this result, or computes a random power of g_1:

$$h'(m) = \begin{cases} h(m) & \text{with probability } P, \\ {g_1}^r & \text{otherwise.} \end{cases}$$

The integer r is selected randomly modulo p. The exposure forger records (m, r) for each hashed message. We determine the optimal value of P when we calculate the exposure forger's success probability. When the replacement forger requests a signature by user i on message m, the exposure forger's reply depends on the hash type of m. If $h'(m)$ equals $h(m)$, then it forwards the signing query to user i's signing oracle. Otherwise, it looks up the value of r for this message and computes the appropriate power of user i's public key:

$$\sigma_i'(m) = \begin{cases} \sigma_i(m) & \text{if } h'(m) = h(m), \\ {x_i}^r & \text{if } h'(m) = {g_1}^r. \end{cases}$$



$(x_1, y_1), \ldots, (x_n, y_n)$
EXPOSURE FORGER
$(x_1, y_1), \ldots, (x_n, y_n)$
H
m
$h(m)$
H
m
$h'(m) = \begin{cases} h(m) & \text{pr. } \frac{1}{m_r+1} \\ {g_1}^r & \text{else} \end{cases}$
S_i
m
$\sigma_i(m)$
S_i
m
$\sigma_i'(m) = \begin{cases} \sigma_i(m) \\ {x_i}^r \end{cases}$
REPLACE-MENT FORGER
S_j
replace (x_j, y_j) with (u_j, v_j)
$\sigma_A' = \sigma_A \cdot \prod_{i \in S_2} {x_{\alpha_i}}^{-r_i} \cdot \prod_{i \in S_4} {u_{\alpha_i}}^{-r_i}$
σ_A, $m_1, \ldots, m_k$, and $\alpha_1, \ldots, \alpha_k$
σ_A', $\{m_i\}_{i \in S_1}$, and $\{\alpha_i\}_{i \in S_1}$

Figure 4.8: The reduction from exposure forgery to replacement forgery has a tightness gap of m_r.

These signatures are valid. First, when $h'(m) = h(m)$, the signature is valid because it was obtained from a real signing oracle:

$$e(\sigma_i'(m), g_2) = e(\sigma_i(m), g_2) = e(h(m), y_i) = e(h'(m), y_i).$$

When $h'(m)$ is a random power of g_1, the signature is valid due to properties of the pairing:

$$e(\sigma_i'(m), g_2) = e({x_i}^r, g_2) = e({g_1}^{z_i \cdot r}, g_2) = e({g_1}^r, {g_2}^{z_i}) = e(h'(m), y_i).$$

At any time, the replacement forger can replace user j's public key with any (u_j, v_j) in $G_1 \times G_2$. The exposure forger keeps track of these replaced keys by recording j and (u_j, v_j). Let $(x_1', y_1'), \ldots, (x_n', y_n')$ denote the public keys at the end of the algorithm:

$$(x_i', y_i') = \begin{cases} (u_i, v_i) & \text{if the replacement forger replaced user } i\text{'s public key,} \\ (x_i, y_i) & \text{otherwise.} \end{cases}$$

After time t, the replacement forger succeeds with some probability. It outputs a valid aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, and k user indices $\alpha_1, \ldots, \alpha_k$, for some positive integer k that is at most n. The forged signature satisfies the equation $e(\sigma_A, g_2) = \prod_{i=1}^{k} e(h'(m_i), y'_{\alpha_i})$ and the replacement forger did not replace user α_1's public key, nor did it request a signature by this user on m_1.

Partition the integers from 1 to k into the following four sets:

$$\begin{aligned}
S_1 &= \{i \in [k] \mid (x'_{\alpha_i}, y'_{\alpha_i}) = (x_{\alpha_i}, y_{\alpha_i}) \text{ and } h'(m_i) = h(m_i)\},\\
S_2 &= \{i \in [k] \mid (x'_{\alpha_i}, y'_{\alpha_i}) = (x_{\alpha_i}, y_{\alpha_i}) \text{ and } h'(m_i) = {g_1}^{r_i}\},\\
S_3 &= \{i \in [k] \mid (x'_{\alpha_i}, y'_{\alpha_i}) = (u_{\alpha_i}, v_{\alpha_i}) \text{ and } h'(m_i) = h(m_i)\}, \text{ and}\\
S_4 &= \{i \in [k] \mid (x'_{\alpha_i}, y'_{\alpha_i}) = (u_{\alpha_i}, v_{\alpha_i}) \text{ and } h'(m_i) = {g_1}^{r_i}\}.
\end{aligned}$$

In particular, we know that 1 is in S_1 or S_2 since user α_1's key was not replaced. The exposure forger succeeds if and only if the following events occur:

(E_1) The replacement forger succeeds after time t. From the replacement forger's point of view, the exposure forger is indistinguishable from a real hashing oracle and signing oracle since the message hashes are random and the signatures are valid. Therefore, the replacement forger outputs a valid forgery with probability at least ϵ.

(E_2) The hash of m_1 is not a random power of g_1 and for each integer i from 2 to k, if the replacement forger replaced user α_i's public key, then the message m_i has a hash that is a random power of g_1. That is, the index 1 is in S_1 and S_3 is empty. Since we know that 1 is either in S_1 or S_2 and the replacement forger can replace at most m_r public keys, the probability of this event given the first event is $P \cdot (1-P)^{m_r}$.

The exposure forger's success probability ϵ' is at least $\Pr(E_1 \wedge E_2) = \epsilon \cdot P \cdot (1-P)^{m_r}$. By Equation (1.1), the value $P = 1/(m_r + 1)$ maximizes this lower bound on the exposure forger's success probability. Finally, applying the approximation of e^{-1} in Equation (1.2) gives $\epsilon' \geq \epsilon/(e \cdot (m_r + 1))$.

If all three events occur, then the exposure forger computes the following forged signature in G_1:

$$\sigma'_A = \sigma_A \cdot \prod_{i \in S_2} {x_{\alpha_i}}^{-r_i} \cdot \prod_{i \in S_4} {u_{\alpha_i}}^{-r_i}.$$

It outputs σ'_A, the user indices α_i for i in S_1, and the messages m_i for i in S_1.

This signature is valid since

$$\begin{aligned}
e(\sigma'_A, g_2) &= e\left(\sigma_A, g_2\right) \cdot \prod_{i \in S_2} e\left({x_{\alpha_i}}^{-r_i}, g_2\right) \cdot \prod_{i \in S_4} e\left({u_{\alpha_i}}^{-r_i}, g_2\right) \\
&= \prod_{i=1}^{k} e(h'(m_i), y'_{\alpha_i}) \cdot \prod_{i \in S_2} e\left({g_1}^{-r_i \cdot z_{\alpha_i}}, g_2\right) \cdot \prod_{i \in S_4} e\left({g_1}^{-r_i \cdot w_{\alpha_i}}, g_2\right) \\
&= \prod_{i=1}^{k} e(h'(m_i), y'_{\alpha_i}) \cdot \prod_{i \in S_2} e\left(h'(m_i)^{-1}, y_{\alpha_i}\right) \cdot \prod_{i \in S_4} e\left(h'(m_i)^{-1}, v_{\alpha_i}\right) \\
&= \prod_{i \in S_1} e(h(m_i), y_{\alpha_i}).
\end{aligned}$$

The exposure forger's running time t' is at most $t + (q_h + q_s + n - 1) \cdot T_e + (n-1) \cdot T_m$. It makes at most as many hashing queries as the replacement forger, q_h, and at most as many signing queries, q_s, as the replacement forger. It does not expose any of the private keys corresponding to the n challenge public keys. Hence, given a $(t, \epsilon, q_h, q_s, m_r)$-replacement forger, we can create a $(t', \epsilon', q_h, q_s, m_e)$-exposure forger, where m_r and m_e are any positive integers at most $n - 1$,

$$\begin{aligned}
t' &= t + (q_h + q_s + n - 1) \cdot T_e + (n-1) \cdot T_m, \text{ and} \\
\epsilon' &= \frac{\epsilon}{e \cdot (m_r + 1)}.
\end{aligned}$$

If the BGLS aggregate signature scheme is resistant to $(t', \epsilon', q_h, q_s, m_e)$-exposure forgery, then it is resistant to $(t, \epsilon, q_h, q_s, m_r)$-replacement forgery for values of t and ϵ satisfying the above inequalities. □

The previous reduction is not tight: in the worst case, it loses tightness by the number of users n. The exposure forger does not use its ability to expose private keys, and we are not aware of a tighter reduction.

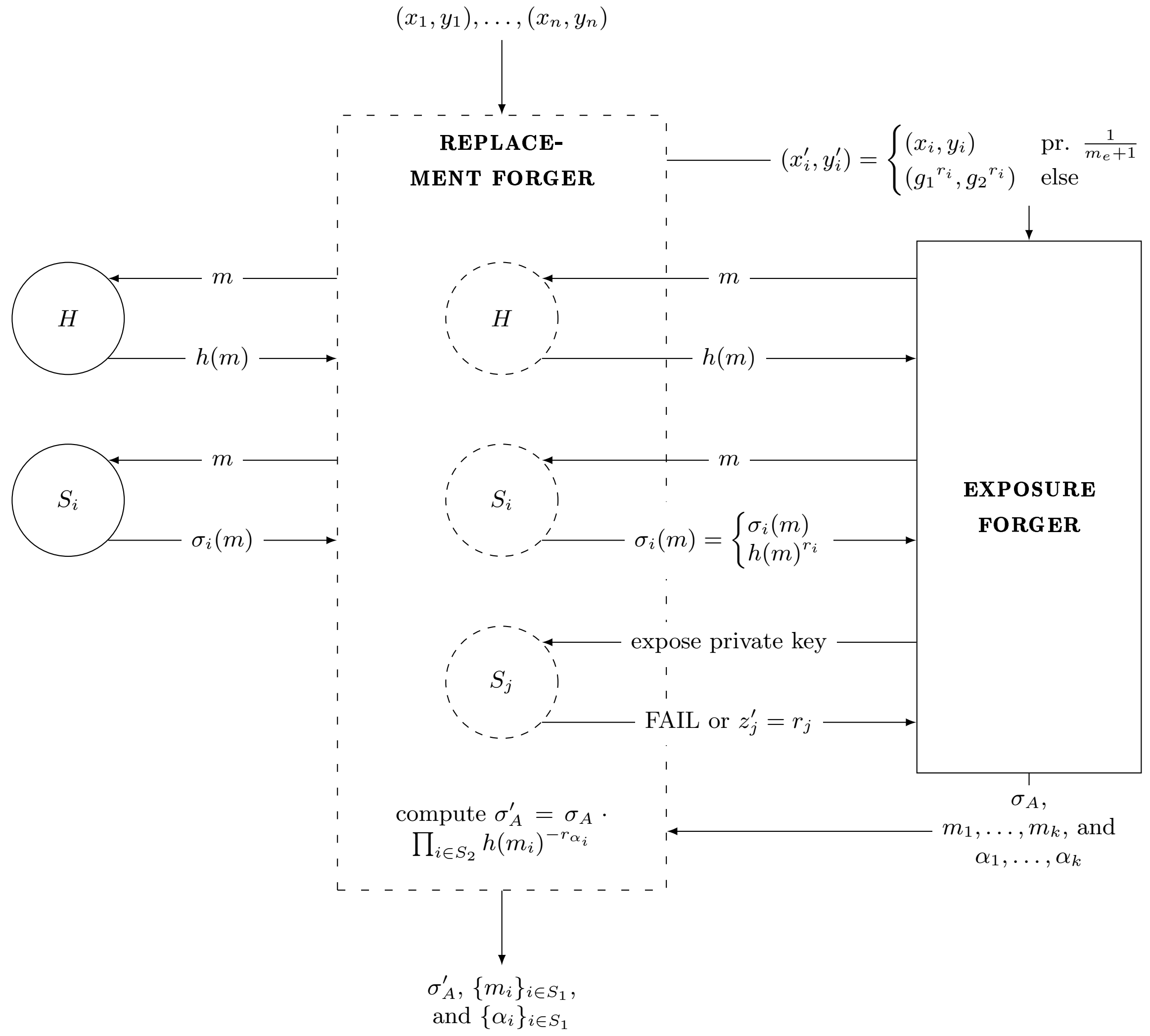
Next, in Theorem 4.6, we devise a reduction from replacement forgery to exposure forgery that has a tightness gap of m_e. The reduction is depicted in Figure 4.9 on page 44.

Theorem 4.6 (replacement forgery $\leq$ exposure forgery). If the BGLS signature scheme in an n-user setting is resistant to $(t', \epsilon', q_h, q_s, m_r)$-replacement forgery, then it is resistant to $(t, \epsilon, q_h, q_s, m_e)$-exposure forgery, for

$$\begin{aligned}
t &= t' - (q_s + 2n - 1) \cdot T_e - (n-1) \cdot T_m, \\
\epsilon &= \epsilon' \cdot e \cdot (m_e + 1),
\end{aligned}$$

and any positive integer m_e that is at most $n - 1$.

Proof. We prove the contrapositive of this statement: we show how to build a replacement forger given an exposure forger. The replacement forger receives n public keys $(x_1, y_1), \ldots, (x_n, y_n)$ and has access to a hashing oracle H and a signing oracle S_i for each



$(x_1, y_1), \ldots, (x_n, y_n)$
REPLACE-MENT FORGER
$(x'_i, y'_i) = \begin{cases} (x_i, y_i) & \text{pr. } \frac{1}{m_e+1} \\ (g_1{}^{r_i}, g_2{}^{r_i}) & \text{else} \end{cases}$
m
H
$h(m)$
m
S_i
$\sigma_i(m)$
H
m
$h(m)$
S_i
m
$\sigma_i(m) = \begin{cases} \sigma_i(m) \\ h(m)^{r_i} \end{cases}$
EXPOSURE FORGER
S_j
expose private key
FAIL or $z'_j = r_j$
compute $\sigma'_A = \sigma_A \cdot \prod_{i \in S_2} h(m_i)^{-r_{\alpha_i}}$
σ_A, $m_1, \ldots, m_k$, and $\alpha_1, \ldots, \alpha_k$
σ'_A, $\{m_i\}_{i \in S_1}$, and $\{\alpha_i\}_{i \in S_1}$

Figure 4.9: The reduction from replacement forgery to exposure forgery has a tightness gap of m_e.

user i. It must give n public keys to the exposure forger. With probability P, it gives the exposure forger the ith public key it received; otherwise it gives the exposure forger a pair of random powers:

$$(x'_i, y'_i) = \begin{cases} (x_i, y_i) & \text{with probability } P, \\ ({g_1}^{r_i}, {g_2}^{r_i}) & \text{otherwise.} \end{cases}$$

The replacement forger stores (i, r_i) for each public key that is a pair of random powers. We determine the optimal value of P when computing the replacement forger's success probability.

When the exposure forger requests the hash of a message m, the replacement forger simply forwards the query to H and returns the result, $h(m)$, to the exposure forger. When the exposure forger requests a signature from user i on message m, the replacement forger either forwards the query to S_i or looks up the integer r_i and computes the appropriate power of $h(m)$:

$$\sigma'_i(m) = \begin{cases} \sigma_i(m) & \text{if } (x'_i, y'_i) = (x_i, y_i), \\ h(m)^{r_i} & \text{otherwise.} \end{cases}$$

At any point, the exposure forger can request the private key z'_i of user i. If user i's public key is one of the replacement forger's challenge public keys, then the reduction fails since the replacement forger cannot answer. However, if user i's public key is a pair of random powers, then the replacement forger looks up the integer r_i and gives it to the exposure forger.

If the exposure forger did not make any private key requests that the replacement forger could not answer, then after time at most t, the exposure forger either fails or outputs a valid forged aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, and k user indices $\alpha_1, \ldots, \alpha_k$ for some positive integer k that is at most n. It did not replace user α_1's public key, nor did it give the replacement forger a signature on m_1. The replacement forger succeeds if and only if the following three events occur:

(E_1) The exposure forger requests only the private keys of users whose public keys are pairs of random powers. Since it can expose at most m_e private keys, this event happens with probability $(1 - P)^{m_e}$.

(E_2) The exposure forger succeeds after time t. Since the exposure forger received random challenge public keys and the replacement forger correctly simulated the hashing oracle and signing oracles, this event happens with probability at least ϵ, given the first event.

(E_3) User α_1's public key is not a pair of random powers. That is, $(x'_{\alpha_1}, y'_{\alpha_1})$ equals $(x_{\alpha_1}, y_{\alpha_1})$. This event happens with probability P.

The replacement forger's probability of success ϵ' is therefore

$$\Pr(E_1 \wedge E_2 \wedge E_3) \geq \epsilon \cdot P \cdot (1 - P)^{m_e}.$$

By Equation (1.1), the value $P = 1/(m_e + 1)$ maximizes this lower bound on the replacement forger's success probability. By applying the approximation in Equation (1.2), the replacement forger's success probability ϵ' is at least $\epsilon/(e \cdot (m_e + 1))$.

When the three events occur, the replacement forger must isolate the part of the signature σ_A that contains messages signed by users with "real" public keys. Partition the integers from 1 to k into the following two sets, based on whether the corresponding user's public key is "real":

$$S_1 = \{i \in [k] \mid (x'_{\alpha_i}, y'_{\alpha_i}) = (x_{\alpha_i}, y_{\alpha_i})\}, \text{ and}$$
$$S_2 = \{i \in [k] \mid (x'_{\alpha_i}, y'_{\alpha_i}) = ({g_1}^{r_{\alpha_i}}, {g_2}^{r_{\alpha_i}})\}.$$

Then, the replacement forger computes the following value:

$$\sigma'_A = \sigma_A \cdot \prod_{i \in S_2} h(m_i)^{-r_{\alpha_i}}.$$

The replacement forger outputs the aggregate signature σ'_A, the messages m_i for each i in S_1, and the user indices α_i for each i in S_1. The forged signature σ'_A is valid since

$$\begin{aligned} e\left(\sigma'_A, g_2\right) &= e\left(\sigma_A, g_2\right) \cdot \prod_{i \in S_2} e\left(h(m_i)^{-r_{\alpha_i}}, g_2\right) \\ &= \prod_{i=1}^{k} e\left(h(m_i), y'_{\alpha_i}\right) \cdot \prod_{i \in S_2} e\left(h(m_i)^{-r_{\alpha_i}}, g_2\right) \\ &= \left(\prod_{i \in S_1} e\left(h(m_i), y_{\alpha_i}\right) \cdot \prod_{i \in S_2} e\left(h(m_i), {g_2}^{r_{\alpha_i}}\right) \right) \cdot \prod_{i \in S_2} e\left(h(m_i), {g_2}^{-r_{\alpha_i}}\right) \\ &= \prod_{i \in S_1} e\left(h(m_i), y_{\alpha_i}\right). \end{aligned}$$

The replacement forger does not replace any public keys. It makes as many hash queries and at most as many signature queries as the exposure forger. Since the exposure forger never requests a signature on m_1, the replacement forger does not either. The replacement forger performs at most $n + q_s + n - 1$ exponentiations and $n - 1$ multiplications. Therefore, the time t' required by the replacement forger is at most $t + (q_s + 2n - 1) \cdot T_e + (n-1) \cdot T_m$.

Hence, given a $(t, \epsilon, q_h, q_s, m_e)$-exposure forger, we can construct a $(t', \epsilon', q_h, q_s, m_r)$-replacement forger, for

$$t' = t + (q_s + 2n - 1) \cdot T_e + (n - 1) \cdot T_m,$$
$$\epsilon' = \frac{\epsilon}{e \cdot (m_e + 1)},$$

and any positive integer m_r that is at most $n - 1$. □

This reduction does not use the replacement capabilities of the replacement forger. We do not know of such a reduction that is tighter, even when the pairing is symmetric.

Chapter 5

A new aggregate forgery problem

The reductions between exposure forgery and replacement forgery are not tight in either direction unless m_r or m_e is very small. At this point, it is clear only that replacement forgery is no harder than targeted-user forgery. We combine the capabilities of the replacement forger and exposure forger to create the replacement-and-exposure forger. Again, we assume that if the replacement-and-exposure forger exposed a user's private key, then its forged aggregate signature will not include a signature on any message by that user.

Definition 5.1.1. A **replacement-and-exposure forger** is an adversary with the following capabilities and goals. It receives n randomly chosen public keys $(x_1, y_1), \ldots, (x_n, y_n)$, corresponding to users 1 to n. It adaptively queries a hashing oracle and n individual signing oracles with messages. At any point, the forger can choose to corrupt user j by exposing its private key z_j or replacing its public key with any element in $G_1 \times G_2$.
The forger's goal is to output k user indices $\alpha_1, \ldots, \alpha_k$, k distinct messages $m_1, \ldots, m_k$, and a valid aggregate signature σ_A comprising user α_i's signature on message m_i, for each integer i from 1 to k, where k is a positive integer at most n. The forger succeeds if it did not expose any of these k users' private keys, it did not replace user α_1's public key, and it did not request a signature on m_1 from user α_1's signing oracle.

Definition 5.1.2. A $(\mathbf{t}, \epsilon, \mathbf{q_h}, \mathbf{q_s}, \mathbf{m_r}, \mathbf{m_e})$**-replacement-and-exposure forger** runs in time at most t, succeeds with probability at least ϵ, makes at most q_h hashing queries and q_s signing queries, replaces at most m_r public keys, and exposes at most m_e private keys. The success probability is computed over all possible inputs $(x_1, y_1), \ldots, (x_n, y_n)$ and all of the forger's coin tosses.

Definition 5.1.3. An aggregate signature scheme is **resistant to** $(\mathbf{t}, \epsilon, \mathbf{q_h}, \mathbf{q_s}, \mathbf{m_r}, \mathbf{m_e})$**-replacement-and-exposure forgery** if no $(t, \epsilon, q_h, q_s, m_r, m_e)$-replacement-and-exposure forger exists.

First, we remark that the reductions from replacement-and-exposure forgery to exposure forgery and replacement forgery are tight since replacement-and-exposure forgers have all the capabilities of these two types of forgers.

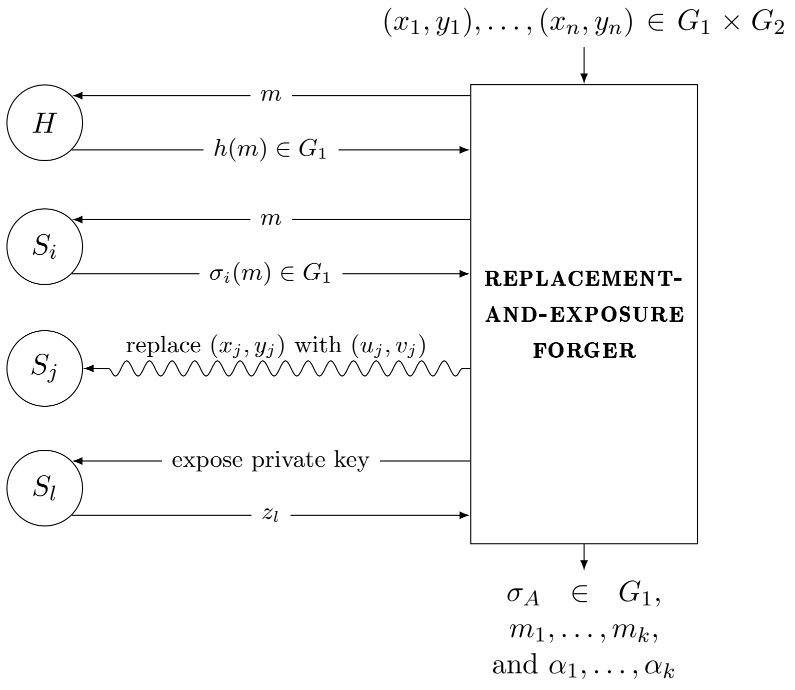

$(x_1, y_1), \ldots, (x_n, y_n) \in G_1 \times G_2$
H
m
$h(m) \in G_1$
S_i
m
$\sigma_i(m) \in G_1$
S_j
replace (x_j, y_j) with (u_j, v_j)
S_l
expose private key
z_l
REPLACEMENT-AND-EXPOSURE FORGER
$\sigma_A \in G_1$, $m_1, \ldots, m_k$, and $\alpha_1, \ldots, \alpha_k$

Figure 5.1: Capabilities and goals of a replacement-and-exposure forger.

Theorem 5.1 (replacement-and-exposure forgery $\leq$ exposure forgery). If the BGLS signature scheme in an n-user setting is resistant to $(t, \epsilon, q_h, q_s, m_r, m_e)$-replacement-and-exposure forgery, then it is also resistant to $(t, \epsilon, q_h, q_s, m_e)$-exposure forgery.

Theorem 5.2 (replacement-and-exposure forgery $\leq$ replacement forgery). If the BGLS signature scheme in an n-user setting is resistant to $(t, \epsilon, q_h, q_s, m_r, m_e)$-replacement-and-exposure forgery, then it is also resistant to $(t, \epsilon, q_h, q_s, m_r)$-replacement forgery.

Next, we examine reductions in the other direction. Will the reduction from exposure forgery to replacement-and-exposure forgery tell us more about the relative hardness of replacement forgery and exposure forgery? This reduction, in the proof of Theorem 5.3, is depicted in Figure 5.2 on page 49.

Theorem 5.3 (exposure forgery $\leq$ replacement-and-exposure forgery). If the BGLS signature scheme in an n-user setting is resistant to $(t', \epsilon', q_h, q_s, m_e)$-exposure forgery, then it is also resistant to $(t, \epsilon, q_h, q_s, m_r, m_e)$-replacement-and-exposure forgery, for any positive integer m_r at most $n-1$,

$$t = t' - (q_h + q_s + n - 1) \cdot T_e - (n-1) \cdot T_m, \text{ and}$$
$$\epsilon = \epsilon' \cdot e \cdot (m_r + 1).$$

Proof. We prove the contrapositive: we show how to build an exposure forger given a replacement-and-exposure forger. The exposure forger receives n challenge public keys

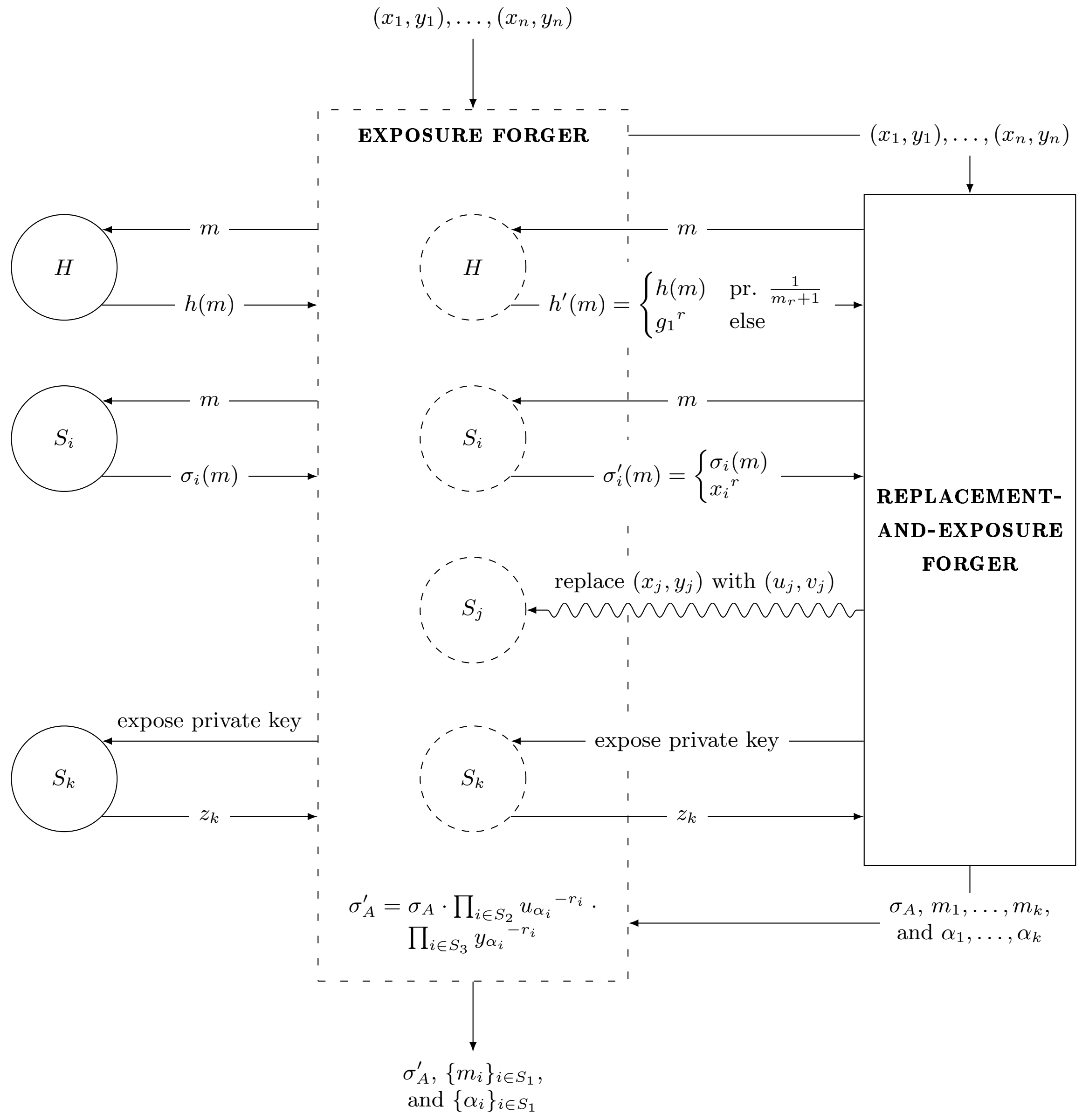

$(x_1, y_1), \ldots, (x_n, y_n)$
EXPOSURE FORGER
$(x_1, y_1), \ldots, (x_n, y_n)$
m
H
$h(m)$
m
H
$h'(m) = \begin{cases} h(m) & \text{pr. } \frac{1}{m_r+1} \\ {g_1}^r & \text{else} \end{cases}$
m
S_i
$\sigma_i(m)$
m
S_i
$\sigma_i'(m) = \begin{cases} \sigma_i(m) \\ {x_i}^r \end{cases}$
REPLACEMENT-AND-EXPOSURE FORGER
S_j
replace (x_j, y_j) with (u_j, v_j)
expose private key
S_k
z_k
expose private key
S_k
z_k
$\sigma_A' = \sigma_A \cdot \prod_{i \in S_2} {u_{\alpha_i}}^{-r_i} \cdot \prod_{i \in S_3} {y_{\alpha_i}}^{-r_i}$
σ_A, $m_1, \ldots, m_k$, and $\alpha_1, \ldots, \alpha_k$
σ_A', $\{m_i\}_{i \in S_1}$, and $\{\alpha_i\}_{i \in S_1}$

Figure 5.2: The reduction from exposure forgery to replacement-and-exposure forgery has a tightness gap of m_r.

$(x_1, y_1), \ldots, (x_n, y_n)$ and has access to a hashing oracle H and n individual signing oracles S_i. It can expose at most m_e of the users' private keys.

First, the exposure forger gives the replacement-and-exposure forger the same n public keys. When the replacement-and-exposure forger requests the hash of a message, the exposure forger either forwards the query to the hashing oracle, or computes a random power of g_1:

$$h'(m) = \begin{cases} h(m) & \text{with probability } P, \\ {g_1}^r & \text{otherwise.} \end{cases}$$

In the second case, the exposure forger records (m, r), where r is the randomly chosen integer modulo p. We determine the optimal value of P when analyzing the exposure forger's success probability.

When the replacement-and-exposure forger requests a signature from user i on a message, the exposure forger either forwards the query to user i's signing oracle, or computes the appropriate power of x_i:

$$\sigma_i'(m) = \begin{cases} \sigma_i(m) & \text{if } h'(m) = h(m), \\ {x_i}^r & \text{if } h'(m) = {g_1}^r. \end{cases}$$

In the first case, the signature is clearly valid. When the hash is a random power of g_1,

$$e\left(\sigma_i'(m), g_2\right) = e\left({x_i}^r, g_2\right) = e\left({g_1}^{w_i \cdot r}, g_2\right) = e\left(h'(m), y_i\right),$$

so the signature is also valid in the second case.

The replacement-and-exposure forger can, at any time, replace user j's public key with any public key (u_j, v_j) in $G_1 \times G_2$. The exposure forger keeps track of these replaced keys by recording j and (u_j, v_j). At any point, the replacement-and-exposure forger can query user k's oracle for its private key, which the exposure forger then obtains by querying the real user k's oracle.

After time at most t and with some probability, the replacement-and-exposure forger outputs a valid forged aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, and k user indices $\alpha_1, \ldots, \alpha_k$ for some positive integer k that is at most n. The replacement-and-exposure forger did not expose the private key of any user α_i, nor did it replace the public key of user α_1 or request a signature on m_1 from it.

The exposure forger succeeds if and only if the following events occur:

(E_1) The replacement-and-exposure forger succeeds in time t. Since the replacement-and-exposure forger receives random challenge public keys, the exposure forger correctly simulates the hashing oracle and signing oracles, and the exposure forger is always able to reveal the requested private keys, this event happens with probability at least ϵ.

(E_2) The hash of message m_1 is not a random power of g_1 and for each integer i from 2 to k, if user α_i's public key was replaced, then the hash of m_i is a random power of g_1. That is, $h'(m_1) = h(m_1)$ and $h'(m_i) = {g_1}^{r_i}$ for all i such that $(x_{\alpha_i}, y_{\alpha_i})$ was replaced. Given the first event, this event happens with probability $\Pr(E_2 \mid E_1) \geq P \cdot (1-P)^{m_r}$.

The exposure forger's probability of success ϵ' is therefore

$$\Pr(E_1 \wedge E_2) \geq \epsilon \cdot P \cdot (1-P)^{m_r}.$$

By Equation (1.1), the value $P = 1/(m_r + 1)$ maximizes this lower bound on the exposure forger's success probability. By applying the approximation in Equation (1.2), the exposure forger's success probability ϵ' is at least $\epsilon/(e \cdot (m_r + 1))$.

Let $(x'_1, y'_1), \ldots, (x'_n, y'_n)$ denote the public keys at the end of the algorithm:

$$(x'_i, y'_i) = \begin{cases} (u_i, v_i) & \text{if the replacement-and-exposure forger replaced } (x_i, y_i), \\ (x_i, y_i) & \text{otherwise.} \end{cases}$$

When the required events occur, the exposure forger must isolate the part of the forged signature that contains messages with "real" hashes signed by users whose keys were not replaced. Partition the integers from 1 to k into the following three sets:

$$\begin{aligned} S_1 &= \{i \in [k] \mid (x'_{\alpha_i}, y'_{\alpha_i}) = (x_{\alpha_i}, y_{\alpha_i}) \text{ and } h'(m_i) = h(m_i)\}, \\ S_2 &= \{i \in [k] \mid (x'_{\alpha_i}, y'_{\alpha_i}) = (u_{\alpha_i}, v_{\alpha_i}) \text{ and } h'(m_i) = {g_1}^{r_i}\}, \\ S_3 &= \{i \in [k] \mid (x'_{\alpha_i}, y'_{\alpha_i}) = (x_{\alpha_i}, y_{\alpha_i}) \text{ and } h'(m_i) = {g_1}^{r_i}\}. \end{aligned}$$

Event 3 guarantees that any user whose public key is replaced corresponds to an index in S_2. Also, 1 is in the set S_1 since the forger did not replace user α_1's public key and E_2 guarantees that m_1 has a "real" hash. If all three events happen, then the exposure forger computes the following value in G_1:

$$\sigma'_A = \sigma_A \cdot \prod_{i \in S_2} {u_{\alpha_i}}^{-r_i} \cdot \prod_{i \in S_3} {x_{\alpha_i}}^{-r_i}.$$

The exposure forger outputs σ'_A, the messages m_i for each i in S_1, and the user indices α_i for each i in S_1. This forged signature is valid since

$$\begin{aligned} e\left(\sigma'_A, g_2\right) &= e\left(\sigma_A, g_2\right) \cdot \prod_{i \in S_2} e\left({u_{\alpha_i}}^{-r_i}, g_2\right) \cdot \prod_{i \in S_3} e\left({x_{\alpha_i}}^{-r_i}, g_2\right) \\ &= \prod_{i=1}^{k} e\left(h'(m_i), y'_{\alpha_i}\right) \cdot \prod_{i \in S_2} e\left({g_1}^{-w_{\alpha_i} \cdot r_i}, g_2\right) \cdot \prod_{i \in S_3} e\left({g_1}^{-z_{\alpha_i} \cdot r_i}, g_2\right) \\ &= \prod_{i=1}^{k} e\left(h'(m_i), y'_{\alpha_i}\right) \cdot \prod_{i \in S_2} e\left(h'(m_i), {v_{\alpha_i}}^{-1}\right) \cdot \prod_{i \in S_3} e\left(h'(m_i), {y_{\alpha_i}}^{-1}\right) \\ &= \prod_{i \in S_1} e\left(h'(m_i), y'_{\alpha_i}\right) \\ &= \prod_{i \in S_1} e\left(h(m_i), y_{\alpha_i}\right). \end{aligned}$$

The exposure forger exposes only as many private keys as the replacement-and-exposure forger. It makes at most as many hash queries and signature queries as the replacement-and-exposure forger. Since the replacement-and-exposure forger never requests a signature on m_1 from user α_1, the exposure forger does not either. The exposure forger performs at most $q_h + q_s + n - 1$ exponentiations and $n - 1$ multiplications. Therefore, given a $(t, \epsilon, q_h, q_s, m_r, m_e)$-replacement-and-exposure forger, we can construct a $(t', \epsilon', q_h, q_s, m_e)$-exposure forger, for

$$\begin{aligned} t' &= t + (q_h + q_s + n - 1) \cdot T_e + (n - 1) \cdot T_m, \text{ and} \\ \epsilon' &= \frac{\epsilon}{e \cdot (m_r + 1)}. \end{aligned} \qquad \square$$

The reduction from exposure forgery to replacement-and-exposure forgery has a tightness gap of m_r, just like the reduction from exposure forgery to replacement forgery. Next, we examine the reduction from replacement forgery to replacement-and-exposure forgery in Theorem 5.4. The reduction in this theorem is depicted in Figure 5.3 on page 53.

Theorem 5.4 (replacement forgery $\leq$ replacement-and-exposure forgery). If the BGLS signature scheme in an n-user setting is resistant to $(t', \epsilon', q_h, q_s, m_r)$-replacement forgery, then it is also resistant to $(t, \epsilon, q_h, q_s, m_r, m_e)$-replacement-and-exposure forgery, for any positive integer m_e at most $n - 1$,

$$\begin{aligned} t &= t' - (2n + q_s - 1) \cdot T_e - n \cdot T_m, \text{ and} \\ \epsilon &= \epsilon' \cdot e \cdot (m_e + 1). \end{aligned}$$

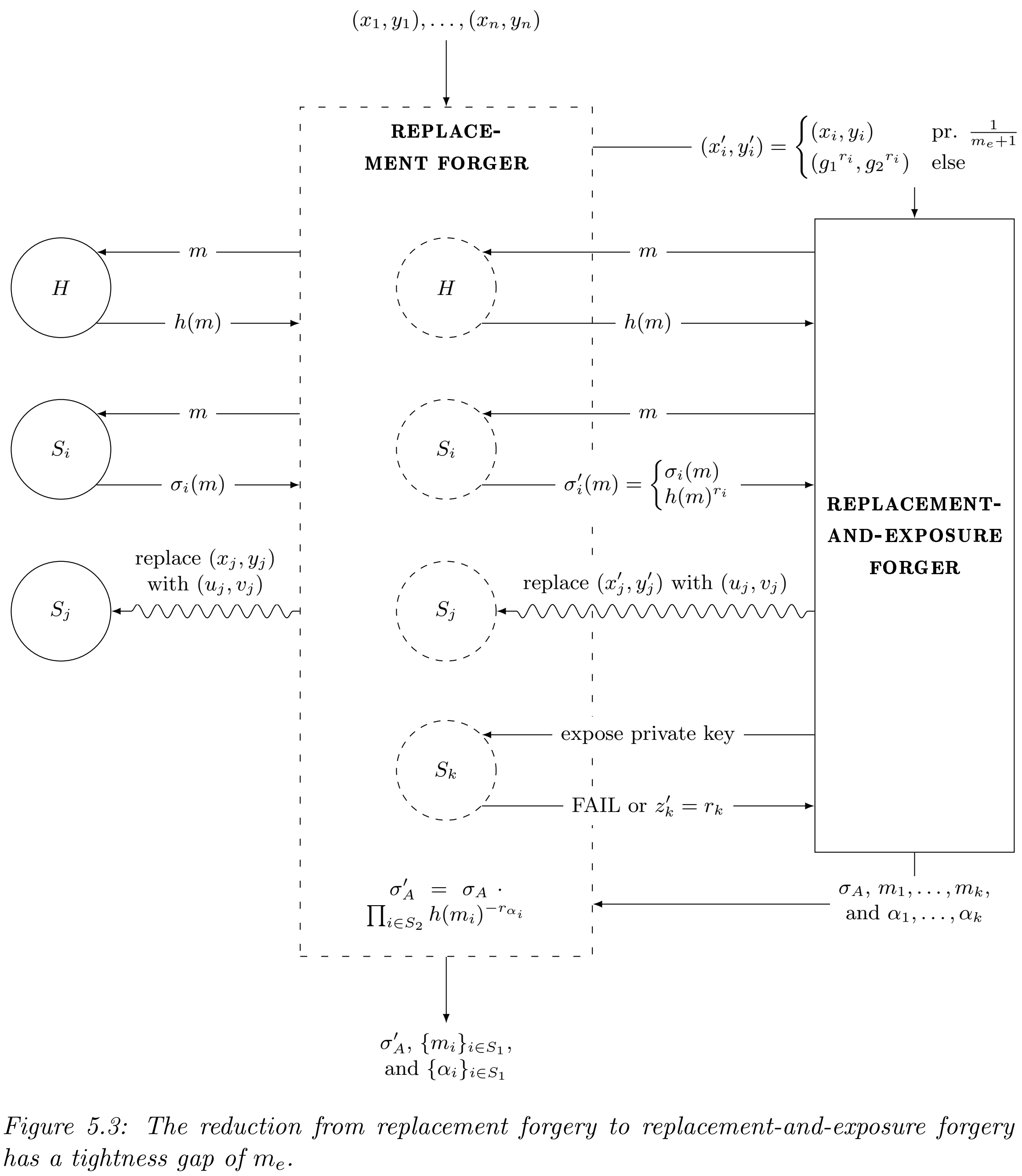
Proof. We prove the contrapositive: we show how to build a replacement forger given a replacement-and-exposure forger. The replacement forger receives n challenge public keys $(x_1, y_1), \ldots, (x_n, y_n)$ and has access to a hashing oracle H and n individual signing oracles S_i. It can replace at most m_r of the users' public keys.

First, the replacement forger must give the replacement-and-exposure forger n public keys. The replacement forger either gives it one of the public keys it received, or computes a pair of random powers:

$$(x_i', y_i') = \begin{cases} (x_i, y_i) & \text{with probability } P, \\ ({g_1}^{r_i}, {g_2}^{r_i}) & \text{otherwise.} \end{cases}$$

In the second case, the replacement forger records (i, r_i), where r_i is the randomly chosen integer modulo p. We determine the optimal value of P when analyzing the replacement forger's success probability.

When the replacement-and-exposure forger requests the hash of a message, the replacement forger simply forwards the query to the hashing oracle and returns the result, $h(m)$ to the replacement-and-exposure forger. When the replacement-and-exposure forger requests user



$(x_1, y_1), \ldots, (x_n, y_n)$
REPLACE-MENT FORGER
$(x'_i, y'_i) = \begin{cases} (x_i, y_i) & \text{pr. } \frac{1}{m_e+1} \\ ({g_1}^{r_i}, {g_2}^{r_i}) & \text{else} \end{cases}$
H
m
$h(m)$
S_i
$\sigma_i(m)$
$\sigma'_i(m) = \begin{cases} \sigma_i(m) \\ h(m)^{r_i} \end{cases}$
REPLACEMENT-AND-EXPOSURE FORGER
S_j
replace (x_j, y_j) with (u_j, v_j)
replace (x'_j, y'_j) with (u_j, v_j)
S_k
expose private key
FAIL or $z'_k = r_k$
$\sigma'_A = \sigma_A \cdot \prod_{i \in S_2} h(m_i)^{-r_{\alpha_i}}$
σ_A, $m_1, \ldots, m_k$, and $\alpha_1, \ldots, \alpha_k$
σ'_A, $\{m_i\}_{i \in S_1}$, and $\{\alpha_i\}_{i \in S_1}$

Figure 5.3: The reduction from replacement forgery to replacement-and-exposure forgery has a tightness gap of m_e.

i's signature on a message, the replacement forger either forwards the query to user i's signing oracle, or computes the appropriate power of $h(m)$:

$$\sigma_i'(m) = \begin{cases} \sigma_i(m) & \text{if } (x_i', y_i') = (x_i, y_i), \\ h(m)^{r_i} & \text{if } (x_i', y_i') = ({g_1}^{r_i}, {g_2}^{r_i}). \end{cases}$$

It is clear that these signatures are both valid.

The replacement-and-exposure forger can, at any time, replace user j's public key (x_j', y_j') with any public key (u_j, v_j) in $G_1 \times G_2$. The replacement forger then also replaces user j's public key (x_j, y_j) with the same (u_j, v_j). The replacement-and-exposure forger can also request the private key z_k' of user k. If user k's public key is one of the replacement forger's challenge public keys, then the reduction fails. Otherwise, the replacement forger gives it the private key $z_k' = r_k$.

After time at most t and with some probability, the replacement-and-exposure forger outputs a valid forged aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, and k user indices $\alpha_1, \ldots, \alpha_k$ for some positive integer k that is at most n. The replacement-and-exposure forger did not expose any of the private keys of these k users. It did not replace user α_1's public key, nor did it request a signature on m_1 from this user. The replacement forger succeeds if and only if the following events occur:

(E_1) The replacement forger can answer all exposure queries. That is, if the replacement-and-exposure forger asks for user k's private key, then $(x_k', y_k') = ({g_1}^{r_k}, {g_2}^{r_k})$. This event happens with probability at least $(1-P)^{m_e}$.

(E_2) The replacement-and-exposure forger succeeds after time at most t. Since the keys it receives are random and the replacement forger correctly simulates the hashing oracle and signing oracles, the probability that this event happens given the first event is at least ϵ.

(E_3) User α_1's public key is not a pair of random powers. That is, $(x_{\alpha_1}', y_{\alpha_1}') = (x_{\alpha_1}, y_{\alpha_1})$. Given the first two events, this event happens with probability at least P.

Therefore, the lower bound on the replacement forger's probability of success ϵ' satisfies

$$\Pr(E_1 \wedge E_2 \wedge E_3) \geq \epsilon \cdot P \cdot (1-P)^{m_e}.$$

By Equation (1.1), the value $P = 1/(m_e + 1)$ maximizes this lower bound. By applying the approximation in Equation (1.2), the replacement forger's success probability ϵ' is at least $\epsilon/(e \cdot (m_e + 1))$.

When the three events occur, the replacement forger must isolate the part of the forged signature that contains messages signed by users with "real" public keys. Partition the integers from 1 to k into the following two sets:

$$S_1 = \{i \in [k] \mid v_{\alpha_i}' = v_{\alpha_i} \quad \text{or} \quad v_{\alpha_i}' = w_{\alpha_i}\},$$
$$S_2 = \{i \in [k] \mid v_{\alpha_i}' = {g_2}^{r_{\alpha_i}} \text{ and } v_{\alpha_i}' \text{ is not replaced}\}.$$

Event 3 guarantees that 1 is in S_1 since user α_1's public key is not a pair of random powers. If all three events happen, the replacement forger computes the following value in G_1:

$$\sigma'_A = \sigma_A \cdot \prod_{i \in S_2} h(m_i)^{-r_{\alpha_i}}.$$

The replacement forger then outputs σ'_A, the messages m_i for i in S_1, and the user indices α_i for i in S_1. It replaces only as many public keys as the replacement-and-exposure forger. It makes at most as many hash and signature queries as the replacement-and-exposure forger. Since the replacement-and-exposure forger never requests a signature on m_1 from user α_1, the exposure forger does not either. Finally, the forged signature σ'_A is valid since

$$\begin{aligned} e\left(\sigma'_A, g_2\right) &= e\left(\sigma_A, g_2\right) \cdot \prod_{i \in S_2} e\left(h(m_i^{-r_{\alpha_i}}, g_2\right) \\ &= \prod_{i=1}^{k} e\left(h(m_i), v'_{\alpha_i}\right) \cdot \prod_{i \in S_2} e\left(h(m_i), {v'_{\alpha_i}}^{-1}\right) \\ &= \prod_{i \in S_1} e\left(h(m_i), v'_{\alpha_i}\right). \end{aligned}$$

The indices in S_1 correspond to users whose public keys are challenge public keys that the replacement forger received, or to users whose public keys were replaced by the replacement-and-exposure forger and thus also the replacement forger. The replacement forger performs at most $n + q_s + n - 1$ exponentiations and n multiplications. Therefore, given a $(t, \epsilon, q_h, q_s, m_r, m_e)$-replacement-and-exposure forger, we can construct a $(t', \epsilon', q_h, q_s, m_r)$-replacement forger, for

$$\begin{aligned} t' &= t + (2n + q_s - 1) \cdot T_e + n \cdot T_m, \text{ and} \\ \epsilon' &= \frac{\epsilon}{e \cdot (m_e + 1)}. \end{aligned} \qquad \square$$

Like the reduction from replacement forgery to exposure forgery, the previous reduction from replacement forgery to replacement-and-exposure forgery has a tightness gap of m_e. From the reductions, it is still unclear whether exposure forgery or replacement forgery is easier. However, our goal is to show that BGLS is secure against the easiest problem, and replacement-and-exposure forgery is definitely easier than both exposure forgery and replacement forgery.

5.1 New security reduction for BGLS

In this section, our goal is to find a reduction from the co-CDH* problem to replacement-and-exposure forgery, which is the easiest forgery problem we consider. So far, we have been analyzing the security of BGLS by comparing types of forgeries. We could compose a chain of reductions (cf. Figure 4.1) to get the desired reduction, and multiply the tightness gaps to get its tightness gap:

- The reduction from solving the co-CDH* problem to targeted-user forgery (Figure 3.3) has a tightness gap of $q_s + n$.

- The reduction from targeted-user forgery to replacement forgery has a tightness gap of n.
- The reduction from replacement forgery to replacement-and-exposure forgery has a tightness gap of m_e.

Composing these three reductions, we obtain a reduction from the co-CDH* problem to replacement-and-exposure forgery with a tightness gap of $(q_s + n) \cdot n \cdot m_e$, which is at most $(q_s + n) \cdot n^2$. Similarly, if we choose exposure forgery as the intermediate step between targeted-user forgery and replacement-and-exposure forgery, we obtain a reduction from the co-CDH* problem to replacement-and-exposure forgery that has a tightness gap of $(q_s + n)(n + m_e) \cdot m_r$, which is at most $(q_s + n) \cdot 2n^2$.

However, it may be possible to find a tighter reduction from solving the co-CDH* problem to replacement-and-exposure forgery. For instance, before directly comparing replacement forgery and exposure forgery in Section 4.4, we could deduce the existence of a reduction from replacement forgery to exposure forgery with a tightness gap of $(n - m_r)(n + m_e)$. However, we found a direct reduction from replacement forgery to exposure forgery with a tightness gap of only m_e. All edges in Figure 4.1 satisfy this property: the tightness gap in the direct reduction from problem A to problem B is at most the product of the tightness gaps along any directed path from problem A to problem B.

Our first attempt at a direct reduction from solving the co-CDH* problem to replacement-and-exposure forgery did not finish due to a complex probability analysis. We briefly outline how to build a co-CDH* solver given a replacement-and-exposure forger. The co-CDH* solver receives h and $x = {g_1}^z$ in G_1, and $y = {g_2}^z$ in G_2. It must eventually output h^z in G_1.

It gives the forger the following n public keys, where r_i is a random integer modulo p:

$$(x_i, y_i) = \begin{cases} (x \cdot {g_1}^{r_i}, y \cdot {g_2}^{r_i}) & \text{with probability } P, \\ ({g_1}^{r_i}, {g_2}^{r_i}) & \text{otherwise.} \end{cases}$$

It answers the forger's hash queries in the following way, where s is a random integer modulo p:

$$h(m) = \begin{cases} h \cdot {g_1}^s & \text{with probability } Q, \\ {g_1}^s & \text{otherwise.} \end{cases}$$

When the forger requests a signature from user i on message m, the solver fails if the message's hash is h-dependent and user i's public key is (x, y)-dependent. If the forger requests the private key z_k of user k and user k's public key is (x, y)-dependent, then the solver fails.

If neither of these events happen, then eventually the replacement-and-exposure forger outputs a valid aggregate signature σ_A on messages $m_1, \ldots, m_k$, by users $\alpha_1, \ldots, \alpha_k$ for some positive integer k that is at most n. It did not replace user α_1's public key, nor did it receive a signature on m_1 by this user. Further, none of these k users' private keys were exposed. For the co-CDH* solver to succeed, user α_1's public key must be (x, y)-dependent, message m_1's hash must be h-dependent, and for users α_2 to α_k, if a public key was replaced, then the corresponding message's hash is not h-dependent.

Now, let n_e and n_r represent the actual number of public keys the forger exposed and replaced. These necessary events yield the following lower bound on the co-CDH* solver's success probability ϵ':

$$\epsilon' \geq \epsilon \cdot (1 - PQ)^{q_s} \cdot (1 - P)^{n_e} \cdot (1 - Q)^{n_r} \cdot P \cdot Q.$$

We could not determine how to optimize P and Q without making very loose approximations.

Maple 14 [1] computes the following optimal values of P and Q:

$$P = -\frac{n_r(n_r \cdot Q + Q - 1)}{Q(n_e \cdot n_r + n_r \cdot q_s + n_e) - n_e}, \text{ and}$$

$$Q \text{ is a solution to } 0 = n_r\,(q_s + n_r + 1)\,Q^2 + (-n_r + n_e \cdot n_r + n_e)\,Q - n_e.$$

Evaluating this expression for certain values of q_s, n_e, and n_r suggests that in some cases, the optimal value of P is less than $1/n$. For example, consider an implementation with $n = 2^{20}$, or about 1 million, users. Suppose that a forger can make a total of $q_s = 2^{40}$ signature queries. Then, if the forger exposes at least 2^{19}, or about half, of the users' private keys, the optimal value of P is less than $1/n$. However, the co-CDH* solver would succeed with negligible probability if none of the n public keys it gives to the forger involve ${g_1}^z$ and ${g_2}^z$.

Therefore, we instead try to find a reduction from solving the co-CDH* problem to replacement-and-exposure forgery for a modified version of the BGLS aggregate signature scheme. We use Katz and Wang's idea of prepending a random bit to a message before hashing it [23]. Now, when a forger interacts with a hashing oracle, it gives it a bit b and a message m. When the forger requests a signature from user i, it is user i itself that chooses the bit b and thus chooses which of two hash values to sign.

We must make one small addition to the assumptions listed at the start of Chapter 3. We assume that a forger never requests a signature on the same message twice. This assumption is reasonable when messages have unique signatures; a forger gains no additional information from getting the same signature twice. Now, however, each message has two signatures, so we must additionally assume that a user always chooses the same bit when it signs a particular message. We present the modified scheme, which we denote by BGLS-KW, below.

Signature Scheme 5.5 (BGLS, Katz-Wang variant, with type III pairing).

- **Set-up:** The order-p groups $G_1 = \langle g_1 \rangle$, $G_2 = \langle g_2 \rangle$, and G_T, the full-domain hash function $h : \{0, 1\}^* \to G_1$, and the type III pairing $e : G_1 \times G_2 \to G_T$ are the same as in BGLS.

- **Key generation:** Same as in BGLS.
- **Signing:** To sign the k distinct messages $m_1, \ldots, m_k \in \{0,1\}^*$ with secret keys $x_1, \ldots, x_k \in \mathbb{Z}_p$, compute $\sigma_A = \prod_{i=1}^{k} h(b_i \mid\mid m_i)^{z_i}$ in G_1 where each b_i is a random bit. The same bit b_i is used each time the message m_i is signed by user i. The signature comprises σ_A and the bits $b_1, \ldots, b_k$.
- **Verification:** To verify the signature σ_A on messages $m_1, \ldots, m_k$ by users with public keys $(x_1, y_1), \ldots, (x_k, y_k)$ and chosen random bits $b_1, \ldots, b_k$, verify that $\prod_{i=1}^{k} e(h(b_i \mid\mid m_i), y_i) = e(\sigma_A, g_2)$.

The bits must be transmitted with the signature, which is inefficient, but we examine this scheme only to say something about the security of the original BGLS scheme against replacement-and-exposure forgers. Koblitz and Menezes compared the RSA-FDH signature scheme with its Katz-Wang variant, where a bit is prepended to a message before it is hashed [25]. They argue that it seems "implausible that the Katz-Wang system could be appreciably more secure than the original RSA signature with full-domain hash function." Similarly, we believe that the reduction from solving co-CDH* to targeted-user forgery with the original BGLS scheme is not less tight than the reduction with the Katz-Wang variant. The reduction is depicted in Figure 5.4 on page 59.

Theorem 5.6 (**Security of BGLS-KW with type III pairing**)**.** If the co-CDH* problem is (t', ϵ')-hard, then the BGLS-KW aggregate signature scheme is resistant to $(t, \epsilon, q_h, q_s, m_r, m_e)$-replacement-and-exposure forgery, for

$$t = t' - (3n + q_h + q_s) \cdot T_e - (4n + q_h) \cdot T_m, \text{ and}$$
$$\epsilon = \epsilon' \cdot 2e^2 \cdot (m_e + 1)(m_r + 1).$$

Proof. We prove the contrapositive of this statement: we show how to build a co-CDH* solver given a replacement-and-exposure forger. The co-CDH* solver receives h and $x = {g_1}^z$ in G_1, and $y = {g_2}^z$ in G_2. It must eventually output h^z in G_1. First, it must give n public keys to the replacement-and-exposure forger. With probability P, it gives the replacement-and-exposure forger a public key that depends on (x, y); otherwise it gives the forger a pair of random powers:

$$(x_i, y_i) = \begin{cases} (x \cdot {g_1}^{r_i}, y \cdot {g_2}^{r_i}) & \text{with probability } P, \\ ({g_1}^{r_i}, {g_2}^{r_i}) & \text{otherwise.} \end{cases}$$

For each public key, the solver records (i, r_i), where r_i is the randomly chosen integer modulo p, and whether it depends on (x, y). We will determine the optimal value of P when computing the co-CDH* solver's success probability.

When the replacement-and-exposure forger requests the hash of the bit b and message m, the solver first selects a random bit b'_m, if it has not already done so for that message. It records (m, b'_m) and gives the forger one of two hashes:

$$h(b \mid\mid m) = \begin{cases} h^{b \oplus b'_m} \cdot {g_1}^s & \text{with probability } Q, \\ {g_1}^s & \text{otherwise,} \end{cases}$$

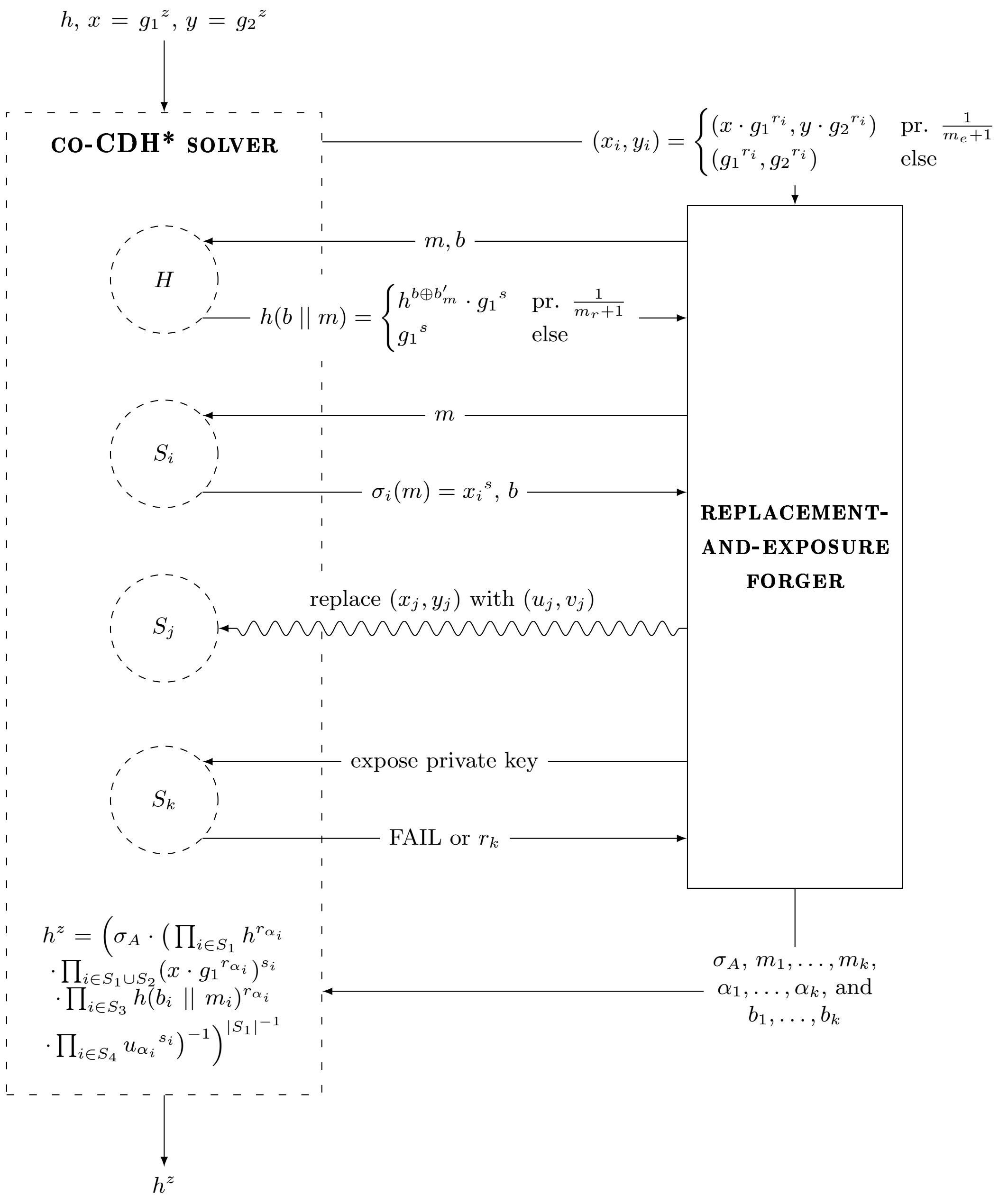

h, $x = {g_1}^z$, $y = {g_2}^z$
CO-CDH* SOLVER
$(x_i, y_i) = \begin{cases} (x \cdot {g_1}^{r_i}, y \cdot {g_2}^{r_i}) & \text{pr. } \frac{1}{m_e+1} \\ ({g_1}^{r_i}, {g_2}^{r_i}) & \text{else} \end{cases}$
m, b
H
$h(b \,||\, m) = \begin{cases} h^{b \oplus b'_m} \cdot {g_1}^s & \text{pr. } \frac{1}{m_r+1} \\ {g_1}^s & \text{else} \end{cases}$
m
S_i
$\sigma_i(m) = {x_i}^s$, b
REPLACEMENT-AND-EXPOSURE FORGER
replace (x_j, y_j) with (u_j, v_j)
S_j
expose private key
S_k
FAIL or r_k
$h^z = \Big(\sigma_A \cdot \big(\prod_{i \in S_1} h^{r_{\alpha_i}} \cdot \prod_{i \in S_1 \cup S_2} (x \cdot {g_1}^{r_{\alpha_i}})^{s_i} \cdot \prod_{i \in S_3} h(b_i \,||\, m_i)^{r_{\alpha_i}} \cdot \prod_{i \in S_4} {u_{\alpha_i}}^{s_i}\big)^{-1}\Big)^{|S_1|^{-1}}$
σ_A, $m_1, \ldots, m_k$, $\alpha_1, \ldots, \alpha_k$, and $b_1, \ldots, b_k$
h^z

Figure 5.4: The reduction from solving the co-CDH problem to replacement-and-exposure forgery in the Katz-Wang variant of BGLS has a tightness gap of about n^2.*

where s is a random integer modulo p. This construction guarantees that at most one of a message's two hashes can include h, and that the bit corresponding to the potentially h-dependent hash is random. The solver also records (m, b, s) for each hashed message.

When the forger requests a signature from user i on message m, the solver always "chooses" to sign a hash that does not involve h. First, suppose that m was hashed only once, it was hashed with $b \neq b'_m$, and its hash is h-dependent. In this case, the solver itself hashes m with $b = b'_m$. Suppose that it chooses s as the random power of g_1. Then, it returns $\sigma_i(m) = {x_i}^s$ and the bit b'_m to the forger. Next, suppose that m has at least one h-independent hash. The solver looks up a value of s that it used to compute this hash, say with bit b. It returns $\sigma_i(m) = {x_i}^s$ and b to the forger. In either case, the signature is valid since ${x_i}^s = ({g_1}^{\log_{g_1} x_i})^s = h(m)^{\log_{g_1} x_i}$.

At any time, the replacement-and-exposure forger can replace user j's public key with any (u_j, v_j) in $G_1 \times G_2$. The solver keeps track of these replaced keys by recording j and (u_j, v_j). The forger can also request the private key z_k of user k. If user k's public key is (x, y)-dependent, then the solver fails since it cannot answer. Otherwise, the co-CDH* solver looks up r_k and gives it to the forger.

Suppose the replacement-and-exposure forger did not make any private key requests that the solver could not answer. Then, after time at most t, the forger either fails or outputs a valid aggregate signature σ_A, k distinct messages $m_1, \ldots, m_k$, k user indices $\alpha_1, \ldots, \alpha_k$, and k bits $b_1, \ldots, b_k$ for some positive integer k that is at most n. User α_1's public key was not replaced, nor did the forger receive a signature on m_1 by this user. Further, none of these k users' private keys were exposed.

The co-CDH* solver succeeds if and only if the following events occur:

(E_1) The forger requests only the private keys of users whose public keys are pairs of random powers. This event happens with probability at least $(1 - P)^{m_e}$.

(E_2) The forger succeeds after time t. Since the forger receives random challenge public keys and the solver correctly simulates the hashing oracle and signing oracles, given the first event, this event happens with probability at least ϵ.

(E_3) The bit b_1 does not equal b'_{m_1}. Since the forger does not know the value of b' for any message, this event happens with probability $1/2$ given the first two events.

(E_4) The hash of $b_1 \mid\mid m_1$ is h-dependent. Given the previous events, this event happens with probability Q.

(E_5) User α_1's public key is (x, y)-dependent. Given the previous events, this event happens with probability P.

(E_6) For users α_2 to α_k, if a public key was replaced, then the corresponding message's hash is not h-dependent. Given the previous events, this event happens with probability at least $(1 - Q)^{m_r}$.

The solver's probability of success ϵ' is therefore at least

$$\Pr(E_1 \wedge \cdots \wedge E_6) \geq (1-P)^{m_e} \cdot \epsilon \cdot \frac{1}{2} \cdot Q \cdot P \cdot (1-Q)^{m_r}.$$

By Equation (1.1), the optimal values of P and Q are $P = 1/(m_e+1)$ and $Q = 1/(m_r+1)$. Next, by applying the approximation in Equation (1.2), the solver's success probability ϵ' is at least

$$\frac{\epsilon}{2e^2 \cdot (m_e+1)(m_r+1)}.$$

When the six required events occur, the solver must extract h^z from the signature σ_A. Partition the integers from 1 to k into the following sets, based on the type of key and type of message hash:

$S_1 = \{i \in [k] \mid$ public key is (x,y)-dependent, not replaced; $h(b_i \mid\mid m_i)$ is h-dependent$\}$,
$S_2 = \{i \in [k] \mid$ public key is (x,y)-dependent, not replaced; $h(b_i \mid\mid m_i)$ is not h-dependent$\}$,
$S_3 = \{i \in [k] \mid$ public key is random, not replaced$\}$,
$S_4 = \{i \in [k] \mid$ public key was replaced; $h(b_i \mid\mid m_i)$ is not h-dependent by $E_6\}$.

Then, since there is a unique aggregate signature on these messages by these users with these bits, it must satisfy the following equation:

$$\begin{aligned}
\sigma_A &= \prod_{i\in S_1} h(b_i \mid\mid m_i)^{z_{\alpha_i}} \cdot \prod_{i\in S_2} h(b_i \mid\mid m_i)^{z_{\alpha_i}} \cdot \prod_{i\in S_3} h(b_i \mid\mid m_i)^{z_{\alpha_i}} \cdot \prod_{i\in S_4} h(b_i \mid\mid m_i)^{w_{\alpha_i}} \\
&= \prod_{i\in S_1} (h \cdot {g_1}^{s_i})^{z+r_{\alpha_i}} \cdot \prod_{i\in S_2} ({g_1}^{s_i})^{z+r_{\alpha_i}} \cdot \prod_{i\in S_3} h(b_i \mid\mid m_i)^{r_{\alpha_i}} \cdot \prod_{i\in S_4} ({g_1}^{s_i})^{w_{\alpha_i}} \\
&= \prod_{i\in S_1} h^z \cdot h^{r_{\alpha_i}} \cdot {g_1}^{s_i(z+r_{\alpha_i})} \cdot \prod_{i\in S_2} {g_1}^{s_i(z+r_{\alpha_i})} \cdot \prod_{i\in S_3} h(b_i \mid\mid m_i)^{r_{\alpha_i}} \cdot \prod_{i\in S_4} {u_{\alpha_i}}^{s_i} \\
&= h^{z\cdot|S_1|} \cdot \prod_{i\in S_1} h^{r_{\alpha_i}} \cdot \prod_{i\in S_1\cup S_2} (x \cdot {g_1}^{r_{\alpha_i}})^{s_i} \cdot \prod_{i\in S_3} h(b_i \mid\mid m_i)^{r_{\alpha_i}} \cdot \prod_{i\in S_4} {u_{\alpha_i}}^{s_i}.
\end{aligned}$$

Then, the solver computes the following value, which equals h^z:

$$\left(\sigma_A \cdot \left(\prod_{i\in S_1} h^{r_{\alpha_i}} \cdot \prod_{i\in S_1\cup S_2} (x \cdot {g_1}^{r_{\alpha_i}})^{s_i} \cdot \prod_{i\in S_3} h(b_i \mid\mid m_i)^{r_{\alpha_i}} \cdot \prod_{i\in S_4} {u_{\alpha_i}}^{s_i}\right)^{-1}\right)^{|S_1|^{-1} \bmod p}.$$

The co-CDH* solver performs at most $3n + q_h + q_s$ exponentiations and $4n + q_h$ multiplications. Hence, given a $(t, \epsilon, q_h, q_s, m_r, m_e)$-replacement-and-exposure forger, we can construct a (t', ϵ')-co-CDH* solver, for

$$\begin{aligned}
t' &= t + (3n + q_h + q_s) \cdot T_e + (4n + q_h) \cdot T_m, \text{ and} \\
\epsilon' &= \frac{\epsilon}{2e^2 \cdot (m_e+1)(m_r+1)}.
\end{aligned}$$

□

When m_r and m_e have their maximum values, $n-1$, the reduction loses tightness by a factor of about n^2, and we expect the reduction from the co-CDH* problem to BGLS forgery to have a similar tightness gap. The original security reduction for BGLS, Theorem 3.5 on page 20, already had a tightness gap of $n + q_s$. Is this further loss of tightness significant?

Chapter 6

Conclusion

This thesis addresses a deficiency in the chosen-key security model of BGLS: we believe that it is not realistic to assume that an attacker is given a public key to target. We suggested improvements to the security model for general, non-identity based aggregate signature schemes. We gradually constructed a security model where attackers can choose the target users and can replace other users' public keys or expose their private keys. We presented reductions among the original targeted-user forgery, intermediate notions of forgery, and the stronger notion of replacement-and-exposure forgery. Finally, we gave a security reduction for a variant of the BGLS aggregate signature scheme with respect to replacement-and-exposure forgery and compared its tightness to the original BGLS security reduction. We considered only type III pairings and we supplemented each reduction with a reduction diagram.

Our analyses assumed that there is an upper bound on the total number of signature queries a forger can make. It is not clear whether this assumption is valid. The bound could also be on the number of signature queries it can make per user. Additionally, there could be a limit on the number of signatures in each aggregate signature.

In Subsection 3.2.1, we described a rogue key attack and three ways to prevent it: requiring proof of knowledge of the private key, proof of possession of the private key, or pairwise distinct messages. We chose the last method, so our reductions assume nothing about how users register their public keys. We now briefly consider how the reduction from solving the co-CDH* problem to replacement-and-exposure forgery changes if one of the other two methods is used. Suppose that when the forger replaces some user's public key with (u, v), it must then prove that it knows the corresponding private key w by revealing it to a third party. In the plain public-key model, if the forged signature includes a message signed by a user whose public key was replaced, then that message's hash must be h-independent for the solver to succeed. In this model, however, the co-CDH* solver controls the third party. Thus, it can always remove the part of the forged aggregate signature corresponding to a user whose key was replaced, regardless of the message's hash type. When this type of proof of knowledge is used, the reduction is tighter.

Next, suppose that the forger must prove that it possesses w by signing the hash of (u, v) with BLS, using the current BGLS parameters. In the reduction, the co-CDH* solver controls the hashing oracle. By responding to the hash query with a value that includes h, say h multiplied by a random power of g_1, the co-CDH* solver can compute h^w from the forger's proof-of-possession signature. Again, the reduction is tighter: in the forged signature, the hashes of messages signed by users whose public keys were replaced no longer need to be h-independent. In these two cases, the replacement-and-exposure forger behaves like an exposure forger.

We wonder whether tighter reductions exist from exposure forgery to targeted-user forgery and from replacement forgery to exposure forgery. In the former reduction (Theorem 4.1), the exposure forger does not use its ability to expose any private keys. In the latter reduction (Theorem 4.6), the replacement forger does not use its ability to replace any public keys. In Section 5.1, we presented a security reduction for a variant of BGLS that has a tightness gap of about n^2. The original BGLS security reduction, with a targeted-user forger, has a tightness gap of $n + q_s$. Could there be an attack on BGLS that requires n times less work with a replacement-and-exposure forger than with a targeted-user forger?

Our security reductions are specific to BGLS, so we wonder whether the security models of sequential aggregate signature schemes could also be modified to consider these stronger adversaries. For example, could our reductions apply to the LOSSW aggregate signature scheme [26]? This scheme is pairing-based, like BGLS, but aggregation is sequential and its security reduction does not use random oracles. We wonder whether our reduction techniques would then fail. It would also be interesting to examine whether our reductions could apply to Neven's sequential aggregate signature scheme, which is RSA-based [28].

Many questions surround the tightness of reductions and the significance of non-tight reductions, of which we note a few here. Is there a better way to assess the tightness of a reduction than computing the tightness gap? Why are the storage requirements of reductions not usually analyzed? How much importance should the tightness of a security reduction have when choosing the security parameters of a scheme in practice? What if the non-tightness of a security reduction means the protocol is too inefficient to use in practice? Koblitz and Menezes point out that sometimes, authors "give a non-tight reductionist argument, and [...] give key-length recommendations that would make sense if their proof had been tight" [25].

Giving more power to adversaries results in a security reduction that is even less tight than the original BGLS security reduction. We believe our security model is more comprehensive and realistic, and that in general, it is prudent to honour the careful construction and analysis of a security reduction by addressing its tightness gap when choosing the security parameter in implementations of the scheme.

References

[1] Maple 14. Maplesoft, a division of Waterloo Maple Inc., Waterloo, Ontario. http://www.maplesoft.com, 2010.

[2] A. Bagherzandi and S. Jarecki. Identity-based aggregate and multi-signature schemes based on RSA. In *Public Key Cryptography – PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 480–498, 2010.

[3] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. Cryptology ePrint Archive, Report 2013/400, 2013. http://eprint.iacr.org/2013/400.

[4] M. Bellare, C. Namprempre, and G. Neven. Unrestricted aggregate signatures. In *International Colloqiuum on Automata, Languages and Programming (ICALP)*, pages 411–422, 2007.

[5] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 390–399, 2006.

[6] M. Bellare and G. Neven. Identity-based multi-signatures from RSA. In *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 145–162, 2007.

[7] M. Bellare and P. Rogaway. The exact security of digital signatures—how to sign with RSA and Rabin. In *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, 1996.

[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432, 2003.

[9] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, 2001.

[10] S. Chatterjee, D. Hankerson, E. Knapp, and A. Menezes. Comparing two pairing-based aggregate signature schemes. *Designs, Codes and Cryptography*, 55(2–3):141–167, 2010.

[11] S. Chatterjee, A. Menezes, and P. Sarkar. Another look at tightness. Cryptology ePrint Archive, Report 2011/442, 2011. http://eprint.iacr.org/2011/442.

[12] J.C. Choon and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30, 2003.

[13] J.-S. Coron. On the exact security of Full Domain Hash. In *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–236, 2000.

[14] J.-S. Coron. Optimal security proofs for PSS and other signature schemes. In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287, 2002.

[15] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[16] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23:224–280, 2010.

[17] S.D. Galbraith, K.G. Paterson, and N.P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[18] C. Gentry and R. Zulfikar. Identity-based aggregate signatures. In *Public Key Cryptography – PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 257–273, 2006.

[19] S. Goldwasser, S. Micali, and R.L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[20] F. Göloğlu, R. Granger, G. McGuire, and J. Zumbrägel. On the function field sieve and the impact of higher splitting probabilities: applications to discrete logarithms in $F_{2^{1971}}$ and $F_{2^{3164}}$. In *Advances in Cryptology – CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 109–128, 2013.

[21] A. Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. Cryptology ePrint Archive, Report 2013/095, 2013. http://eprint.iacr.org/2013/095.

[22] S.A. Kakvi and E. Kiltz. Optimal security proofs for Full Domain Hash, revisited. In *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 537–553, 2012.

[23] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *CCS '03: Proceedings of the 10th ACM Conference on Computer and Communications Security*, pages 155–164, 2003.

[24] E. Knapp. On pairing-based signature and aggregate signature schemes. MMath thesis, University of Waterloo, 2008.

[25] N. Koblitz and A. Menezes. Another look at "provable security". *J. Cryptology*, 20:3–37, 2007.

[26] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485, 2006.

[27] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90, 2004.

[28] G. Neven. Efficient sequential aggregate signed data. In *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 52–69, 2008.

[29] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[30] P.C. van Oorschot and M.J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12:1–28, 1999.