New Definitions and Designs for Anonymous Signatures

MIHIR BELLARE[*] SHANSHAN DUAN[†]

Abstract

There is a fundamental tension in society between authenticity and privacy, and finding balances between them is an important line of research. In this paper we investigate a tool called anonymous signatures [12, 21] that is useful in applications like secure auctions. We provide a new formalization and definition of this primitive that lends the primitive more easily and naturally to applications. We then provide numerous schemes meeting our definition. This includes schemes without random oracles and more efficient solutions in the random oracle model. Our schemes are surprisingly cheap in both bandwidth and computation.

Keywords: Signatures, anonymity, hash functions

1 Introduction

There is a fundamental tension in society between authenticity and privacy, and finding balances between them is an important line of research. The approach we will investigate in this paper is anonymous signatures.

What is an anonymous signature? At first hearing, such a thing sounds contradictory, if not impossible. Indeed the natural and desired interpretation of the term is that the signature not reveal the identity (public key) of the signer. But signatures can be verified. So, can't I always identify the signer's public key amongst a list of candidate ones by seeing under which key the signature verifies correctly?

This leads to a few questions. The first is, how can we meaningfully define and achieve anonymous signatures in a way that circumvents the above dilemma? The second is, why should we bother anyway, meaning of what use do we expect anonymous signatures to be? We will address these questions in what follows.

1.1 Previous work

Anonymous signatures were first introduced by Yang, Wong, Deng and Wang (YWDW) [21]. They got around the above-mentioned dilemma by requiring anonymity only when the underlying message is randomly chosen from a large space and is unknown to the verifier. (Which prevents verification.) They envisaged a two-step usage process. In the signing stage, the signer provides the prospective verifier with a signature, but not the message. At this point, the signer's identity remains unknown. In the later opening stage, the signer provides the message (and its public key), so that verification is possible. Fischlin [12] provides some elegant constructions of anonymous signature schemes, meeting the YWDW-definition, without random oracles. His constructions are based on extractors [17, 18].

We point to two drawbacks of the YWDW formulation of anonymous signatures. The first is that it does not lend itself well to the applications which appear to have motivated it. The second is lack of an unambiguity requirement.

[*] Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: `mihir@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/mihir`.

[†] Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: `shduan@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/shduan`.

Class	Scheme	ASign	AVer	—σ—	—κ—	Assumption
RH	RH-BLS	1 exp	1 pr	160	320	CDH
DH	DH-Sch	1 exp	2 exp	160	240	DL
	DH-GQ	1 exp	2 exp	160	2048	Factoring
SP	SP-Sch	1 exp	2 exp	80	160	DL

Figure 1: Summary of anonymous signatures. For each scheme, we show the computational costs of signing and verification, the size of an anonymous signature σ, the size of the de-anonymizer κ and the assumption used to prove security. By "RH" we mean randomized hash. By "DH" we mean deterministic hash. By "SP", we mean splitting. By "exp" we mean an exponentiation. By "pr" we mean a pairing.

The problem from the application perspective is that here messages are certainly not random, and may need to be known in advance to potential verifiers. To illustrate, let us consider secure auctions, which is a canonical application in this area. Alice wishes to place a bid with value bid_A. She wants to be able to claim the bid as hers in case it wins, but otherwise wishes to remain anonymous. The natural solution is to provide, at bidding time, her anonymous signature of bid_A. When the results are announced, and if Alice has won, she can provide the relevant opening information. The difficulty is that, under the YWDW formulation, anonymity in the first stage is only guaranteed if Alice does not provide the message, which in this case is bid_A. However, the auctioneer needs to know the bid in order to determine the winner.

This problem is to some extent recognized in [21, 12]. To solve it, they suggest that the message to be signed be obtained by padding the bid with a random string. Only the random string would be withheld in the first phase. The difficulty is that while this may "work", it moves us outside the YWDW definitional framework, which does not cover such usage and does not give us any guarantees about it. (The explanation for this is somewhat technical. The YWDW definition requires the message to be drawn at random from a message space that is large and fixed beforehand. It is unclear, in this context, how to define this message space, given that the bid may have many possible values, and bids are simply objects chosen by users, rather than ones on which there is some probability distribution.) These difficulties may potentially be resolved by using classes of distributions as per [12], but the alternative definition of anonymous signatures that we will suggest seems to be simpler and more natural.

One might think that these problems are specific to the auction application, but in fact this application is representative. The same issues arise with other applications mentioned in [21] such as anonymous paper review and anonymous key exchange. In summary, the whole "anonymity by message withholding" approach of YWDW just does not seem to map well to applications.

The second weakness of the YWDW definition is that it fails to require what we call *unambiguity*. Namely, given Alice's signature, Bob may be able to produce a public key, different from Alice's, under which the signature verifies, thereby effectively claiming the signature as his own. This means that when Alice's bid wins the auction, Bob can open it, and claim that *he* won the auction! (This does not contradict unforgeability, because the public key Bob provides is different from Alice's.) In fact, we can give specific examples of schemes that meet the YWDW definition but are not unambiguous, meaning are subject to the above attack. Our definition, in contrast, explicitly demands unambiguity, and all our schemes provide it.

1.2 Our notion

We dub our approach "anonymity by partial signature withholding". Instead of denying the verifier the message, we give it the message and deny it part of the signature. Specifically, we ask that any signature produced by the signing algorithm (on input a message M) be a pair of objects (σ, κ). In the signing stage, the potential verifier is provided with the message M and the anonymous signature σ. It is required that she be unable, from these, to determine the identity (public key) of the signer. In the opening stage, the signer provides the de-anonymizer κ (and its public key). Now, the verifier has the full signature and is able to verify.

The application to auctions is immediate and natural. The message is simply Alice's bid bid_A. Alice computes her signature (σ, κ) on this message, and provides the auctioneer with her bid and σ, the anonymous portion of the signature. At this point, her anonymity is guaranteed. Later, if she learns that she has won the auction, she opens by providing the auctioneer with the de-anonymizer κ. Note that the auctioneer is in possession of the full message (bid) from the very beginning, and so, unlike in the message-withholding approach, has all the information necessary to determine the winner of the auction.

We formalize three security properties. Unforgeability requires that it be infeasible to forge Alice's signatures, even when the adversary can obtain Alice's anonymous signatures on messages of her choice and adaptively open any of these she wishes. (This implies standard unforgeability of the full signature.) The anonymity requirement follows [4] and is strong: the adversary cannot distinguish under which of two target public keys an anonymous signature has been created, even when it knows the message and both underlying secret keys. Finally, unambiguity requires that the adversary, given an anonymous signature σ created by Alice, be unable to produce a public key pk_1, a message M_1 and a de-anonymizer κ_1 such that pk_1 is different from Alice's public key and yet (σ, κ) is a valid signature of M_1 under pk_1, even when the adversary can obtain Alice's full signatures under a chosen message attack. Assuming σ was Alice's anonymous signature on her winning bid, unambiguity prevents Bob from being able to open σ under an alternative public key and thereby himself claim the winning bid. Recall that this property was lacking in the YWDW definition of anonymous signatures, opening up auctions based on them to such attacks.

In summary, we have brought a new viewpoint to anonymous signatures, defining them based on partial signature withholding rather than on message withholding. We contend that this change, although simple, is powerful in two ways. The first, which we have already seen, is that we can now cover applications in a natural way. The second, which we will see next, is that we are able to produce natural and practical schemes to meet the definition.

1.3 Schemes and results

Before describing our schemes, let us discuss design criteria, beginning with practical issues and then moving on to theoretical ones.

With regard to efficiency, we wish to minimize both computation and bandwidth. The motivation for the first is obvious. Namely, public key cryptography is already considered expensive in many settings, and we do not wish anonymity to add a further computational burden. Bandwidth deserves more discussion. For wireless devices such as PDAs, cell phones, RFID chips and sensors, battery life is the main limitation. But here, communicating even one bit of data uses significantly more power than executing one 32-bit instruction [2]. Reducing the number of bits to communicate saves power and is important to increase battery life. Also, in many settings, communication is not reliable, and so the fewer the number of bits one has to communicate, the better. For such reasons, we want schemes in which both the anonymous signature and the de-anonymizer are as short as possible.

How well can we hope to do? Any anonymous signature scheme is, of course, a signature scheme. (The anonymous signature and the de-anonymizer together constitute a full signature.) So we cannot hope for computation or bandwidth costs lower than those of standard signature schemes. The issue is to reduce the overhead as much as possible. As we now explain, we do very well.

All our constructions start with a base, standard signature scheme and transform it into an anonymous one. We measure overhead with respect to the base scheme, with the bandwidth overhead being defined as the difference between the length of a full signature in the anonymous scheme and a signature in the base scheme. The computational overhead of our schemes is at most one hash. The bandwidth overhead ranges from 320 bits to (surprisingly) zero bits. In particular, our Schnorr [20] based scheme, SP-ScH, has an 80 bit anonymous signature and a 160 bit de-anonymizer and has zero overhead, in *both* computation and bandwidth.

The primary theoretical issue is to have a proof of security. We provide proofs for all our schemes. We first provide a general construction with a proof without random oracles. We then provide numerous, more efficient, constructions with random oracle model proofs.

Refer to Figure 1 for a summary of the characteristics of our schemes. We now discuss the schemes in more detail.

We first provide a simple, general transform of any signature scheme into an anonymous signature scheme. The transform uses as a tool any commitment scheme. On the theoretical side, this immediately yields constructions without random oracles. (Because standard signature schemes, as well as commitment schemes, without random oracles, are well known.) We contrast this with message-withholding anonymous signatures, where the original work of [21] had no non-random oracle model solutions. The gap was filled by Fischlin [12] using quite sophisticated techniques. In our case (partial signature withholding anonymous signatures) the result is more immediate. On the practical side, we can obtain fairly efficient schemes via a random oracle model instantiation of the commitment scheme, specifically, as a randomized hash. The anonymous signature is the hash of a 160 bit random string together with the base signature, and the de-anonymizer is the base signature together with the random string. We call this the RH construction. The computational overhead is one hash, and the bandwidth overhead is 320 bits. Bandwidth is minimized by choosing BLS [9] as the base signature scheme, and Figure 1 displays the characteristics of the resulting RH-BLS scheme.

We then show how to do better for a class of signature schemes that we call high-entropy schemes. These are schemes where the base signatures are already randomized. In this case, we drop the randomizer introduced above, and set the anonymous signature to merely the hash of the base signature. (The de-anonymizer is simply the base signature.) We provide a direct analysis to prove security. (It doesn't follow from the above-mentioned results). The computational overhead of this DH (deterministic hash) construction is one hash, while the bandwidth overhead has been reduced to 160 bits. What can we use as base schemes? Schemes such as Schnorr [20], GQ [14] and Fiat-Shamir [13] have the desired high entropy. More generally, high entropy is a property of base signature schemes derived from identification protocols via the Fiat Shamir transform [13], so there are numerous other choices as well, all quite efficient. (Note that the BLS scheme does *not* have high entropy and so is unsuitable for use as a base scheme under DH. And, indeed, DH-BLS is insecure.) Figure 1 summarizes the characteristics of the DH-Sch and DH-GQ schemes.

However, we can do even better. In identification-based signature schemes such as that of Schnorr [20], the signature is a pair (σ, κ) where σ is the hash of the commitment (the name given to the first message from the prover) and the message, while κ is the response of the prover when the verifier challenge is σ. We observe that such signature schemes lend themselves very directly to anonymization: we simply use σ as the anonymous signature, and κ as the de-anonymizer. We call this the splitting construction (SP). The result is a scheme that has zero overhead, in both computation and bandwidth! Of course, we need to show that this works. We are able to do this by direct proof based on the general forking lemma of [3]. Observing that the verifier challenge need be only 80 bits long (there are no birthday attacks on the challenge) we obtain the SP-Sch scheme whose characteristics are summarized in Figure 1.

Proving unforgeability of our commitment-based anonymous signature scheme runs into a famous open problem in cryptography called the selective de-commitment problem [11]. The problem is, can an adversary who, given a number of commitments can choose to open some of them, obtain information about the unopened ones? Intuitively not, but nobody has ever been able to prove this, and results in [11] indicate that it is hard. Luckily, in our particular setting, we are able to resolve the problem and prove security of our scheme.

As indicated above, we have shown that one can build an anonymous signature scheme from a commitment scheme. It is natural to ask whether the use of a commitment scheme is necessary. We show that it is. Namely, we show in Section 8 that any anonymous signature scheme can be converted into a commitment scheme. (At the theoretical level there is nothing interesting here since all of these primitives are equivalent to one-way functions [15, 16]. However, our transformation is direct and efficient.)

1.4 Discussion and related work

One might question the motivation for anonymous signatures by suggesting that they are not really necessary for any of the applications we have discussed. For example, for auctions, why not proceed as follows. Alice generates a new public and private key pair for the auction, signs her bid with the newly generated private key, and attaches the public key, together with the signature, to the bid. Since the key pair is only used once, no one can tie Alice to her bid. If she wins the auction, she can reveal the private key used to sign the message, proving that she is the authentic bidder. Similar solutions can be thought up for the other applications such as secure paper review. However, this solution is, in fact, implicitly defining an anonymous signature scheme! Furthermore, this scheme is less efficient than ours. This kind of observation only strengthens the motivation for defining our version of anonymous signatures because it shows that the notion captures intuitive solutions and is thus a natural abstraction in this domain.

A natural question is, how do anonymous signatures (regardless of how they are formulated) differ from group [10, 4] and ring [19, 8] signatures, which also have the goal of providing anonymity? In group signatures, there is a group of users all of which have the same public key, and the signature merely conceals which member of the group is the signer. In a ring signature, each user has its own keys, but computes a signature as a function of the keys of other members of a group, so that the signature does not reveal which member of the group is the signer. In both cases, signatures, once obtained, can immediately be verified. With anonymous signatures, there is no group. Every user has its own key, and computes its signature independently of keys of any other users. However, the signature cannot be verified without the de-anonymization information. The notions are indeed quite different, and have different applications. Also, anonymous signatures, as we have seen, can be implemented much more easily and efficiently than group or ring signatures, which is an advantage.

2 Preliminaries

NOTATION AND CONVENTIONS. We denote by $a_1||\cdots||a_n$ the concatenation of $a_1,\ldots,a_n$. We denote the empty string by ε. Unless otherwise indicated, an algorithm may be randomized. If A is a randomized algorithm then $y \leftarrow\!\$\; A(x_1,\ldots)$ denotes the operation of running A with fresh coins on inputs $x_1,\ldots$ and letting y denote the output. If S is a (finite) set then $s \leftarrow\!\$\; S$ denotes the operation of picking s uniformly at random from S. If $X = x_1||x_2||\ldots||x_n$, then $x_1||x_2||\ldots||x_n \leftarrow X$ denotes the operation of parsing X into its constituents. Similarly, if $X = (x_1, x_2, \ldots, x_n)$ is an n-tuple, then $(x_1, x_2, \ldots, x_n) \leftarrow X$ denotes the operation of parsing X into its elements.

CODE-BASED GAMES. We will use code-based games [7] in definitions and proofs and we recall some background here. A game has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. A game G is executed with an adversary A as follows. First, **Initialize** executes and its outputs are the inputs to A. Then, A executes, its oracle queries being answered by the corresponding procedures of G. When A terminates, its output becomes the input to the **Finalize** procedure. The output of the latter is called the output of the game, and we let G^A denote the event that this game output takes value `true`. Variables not explicitly initialized or assigned are assumed to have value $\perp$, except for booleans which are assumed initialized to `false`. Games G_i, G_j are *identical until bad* if their code differs only in statements that follow the setting of the boolean flag *bad* to true. The following is the Fundamental Lemmas of game-playing:

Lemma 2.1 [7] Let G_i, G_j be identical until *bad* games, and A an adversary. Let BD_i (resp. BD_j) denote the event that the execution of G_i (resp. G_j) with A sets *bad*. Then

$$\Pr\left[\,G_i^A \wedge \mathsf{BD}_i\,\right] = \Pr\left[\,G_j^A \wedge \mathsf{BD}_j\,\right] \text{ and } \Pr\left[\,G_i^A\,\right] - \Pr\left[\,G_j^A\,\right] \leq \Pr\left[\,\mathsf{BD}_j\,\right].$$

```
Initialize
$(pk, sk) \leftarrow\!\!\$\ \mathsf{AKG}()$
$i \leftarrow 0\ ;\ E \leftarrow \emptyset$
Return $pk$

Open($j$)
If $(j \le 0 \vee j > i)$ Return $\perp$
$E \leftarrow E \cup \{M_j\}$
Return $\kappa_j$

ASign($M$)
$i \leftarrow i + 1\ ;\ M_i \leftarrow M$
$(\sigma_i, \kappa_i) \leftarrow\!\!\$\ \mathsf{ASIG}(sk, M_i)$
Return $\sigma_i$

Finalize($M, (\sigma, \kappa)$)
Return $(M \notin E \wedge \mathsf{AVF}(pk, M, (\sigma, \kappa)) = 1)$
```

```
Initialize
$b \leftarrow\!\!\$\ \{0, 1\}$
$(pk_0, sk_0) \leftarrow\!\!\$\ \mathsf{AKG}()$
$(pk_1, sk_1) \leftarrow\!\!\$\ \mathsf{AKG}()$
Return $((pk_0, sk_0), (pk_1, sk_1))$

CH($M$)
$(\sigma, \kappa) \leftarrow\!\!\$\ \mathsf{ASIG}(sk_b, M)$
Return $\sigma$

Finalize($d$)
Return $(b = d)$
```

```
Initialize

Finalize($pk_0, pk_1, M_0, M_1, \sigma, \kappa_0, \kappa_1$)
$d_0 \leftarrow \mathsf{AVF}(pk_0, M_0, (\sigma, \kappa_0))$
$d_1 \leftarrow \mathsf{AVF}(pk_1, M_1, (\sigma, \kappa_1))$
Return $(d_0 = 1 \wedge d_1 = 1 \wedge pk_1 \neq pk_0)$
```

Figure 2: Game AUF-CMA in the left used to define existential unforgeability, game ANON in the center used to define anonymity and game UNAMB in the right used to define unambiguity of anonymous signature scheme $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$.

When we refer to the running time of an adversary A we mean the total time for the execution of G with A where G is the game defining the adversary's advantage. This convention simplifies running time analyses.

DIGITAL SIGNATURES. A digital signature scheme $\mathcal{DS}$ consists of three algorithms with the following functionality. The key generation algorithm SKG returns a pair (pk, sk) of keys consisting of the public key and matching secret key, respectively. The signing algorithm SIG takes the secret key sk and a message M to return a signature s. The deterministic verification algorithm SVF takes a public key pk, a candidate signature s and a message M to return either 1 or 0. We require that all public keys have the same length, as do all signatures output by SIG. The consistency requirement is that for all M we have $\mathsf{SVF}(pk, s, M) = 1$ with probability 1 in the experiment

$$(pk, sk) \leftarrow\!\!\$\ \mathsf{SKG}()\ ;\ s \leftarrow\!\!\$\ \mathsf{SIG}(sk, M).$$

The notion of existential unforgeability is captured by the game EUF-CMA of Figure 8 in Appendix A.

3 Anonymous Signatures

SYNTAX. Syntactically, an anonymous signature scheme $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ is simply a digital signature scheme in which any signature output by the signing algorithm is a pair (σ, κ). We refer to the first component of the pair as the anonymous signature and the second as the de-anonymizer.

SECURITY. We propose three security properties: existential unforgeability, anonymity and unambiguity. The formal definitions are underlain by the games AUF-CMA, ANON and UNAMB shown in Figure 2. The corresponding adversary advantages are defined by $\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(A) = \Pr\left[\text{AUF-CMA}^A_{\mathcal{AS}}\right]$, $\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) = 2 \cdot \Pr\left[\text{ANON}^A_{\mathcal{AS}}\right] - 1$ and $\mathbf{Adv}^{\mathrm{unamb}}_{\mathcal{AS}}(A) = \Pr\left[\text{UNAMB}^A_{\mathcal{AS}}\right]$ respectively.

In game AUF-CMA, an adversary F can query the oracle **ASign** to get an anonymous signature on any message of its choice. It can then, selectively, open whichever of these it pleases, meaning obtain the de-anonymizer, via its **Open** oracle. To win F must output a message M and a valid full signature (σ, κ) of M such that either M was not queried to **ASign** or M was queried to **ASign** but the signature returned

```
Alg $\mathsf{AKG}()$
$(pk, sk) \leftarrow_{\$} \mathsf{SKG}()$
Return $(pk, sk)$
```

```
Alg $\mathsf{ASIG}(sk, M)$
$s \leftarrow_{\$} \mathsf{SIG}(sk, M)$
$(\sigma, \omega) \leftarrow_{\$} \mathsf{CMT}(s||pk)$
$\kappa \leftarrow (s, \omega)$
Return $\sigma$
```

```
Alg $\mathsf{AVF}(pk, M, \sigma, \kappa)$
$(s, \omega) \leftarrow \kappa$
If $(\mathsf{CVF}(\sigma, s||pk, \omega) = 1)$ then
    If $(\mathsf{SVF}(pk, s, M) = 1)$
        then Return 1
Return 0
```

Figure 3: Algorithms defining anonymous signature scheme $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ based on signature scheme $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ and commitment scheme $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$.

was not opened.

The formalization of anonymity follows [4]. The adversary not only gets target public keys pk_0 and pk_1 but also knows the corresponding secret keys sk_0 and sk_1. Via the **CH** oracle, it can obtain an anonymous signature, under sk_b, of a message M of its choice, and it wins if it guesses the challenge bit b. (It is allowed only one query to the **CH** oracle. Security against multiple queries follows by a hybrid argument.)

Suppose Alice has produced an anonymous signature σ of some message M_0 under her public key pk_0. Unambiguity ensures that only Alice can open σ, by requiring that an adversary be unable to produce a public key pk_1, message M_1 and de-anonymizer κ_1 such that $\mathsf{AVF}(pk_1, M_1, (\sigma, \kappa_1)) = 1$ but $pk_0 \neq pk_1$. Actually the requirement is stronger, preventing even Alice herself from priori creating σ which she can later open in two ways. This addresses the concern that Alice may create for herself two identities and, after sending an anonymous signature, "change" the message or identity from which it "originated".

4 The CMT Construction

In this section we propose and prove correct a general transform of any signature scheme into an anonymous one. The idea is simple: the anonymous signature is a commitment to the base signature, and the de-anonymizer is the decommital key together with the base signature. We consider this a good starting point because this simple construction will later be the basis for numerous refinement leading to more efficient schemes. It is also of direct interest because it shows how to achieve anonymous signatures without random oracles and because the proof of unforgeability shows a special case in which we can solve the selective de-commitment problem. We begin by recalling the definition of a commitment scheme.

COMMITMENT SCHEMES. A commitment scheme $\mathcal{CMT}$ consists of two algorithms. The commitment algorithm CMT takes the message M to be committed and returns a pair of (σ, ω) consisting of a commitment σ and decommital key ω. The deterministic verification algorithm CVF takes as input candidate values σ, M, ω of a commital, message and decommital, respectively, and returns either 1 or 0. The consistency requirement is that for all M we have $\mathsf{CVF}(\sigma, M, \omega) = 1$ with probability 1 in the experiemnt $(\sigma, \omega) \leftarrow_{\$} \mathsf{CMT}(M)$. The definitions of hiding and binding are formalized by the games of Figure 8 in Appendix A.

OUR CONSTRUCTION. Our Sign-then-Commit (StC) transform associates to base digital signature scheme $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ and base commitment scheme $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ the anonymous signature scheme $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ whose constituent algorithms are defined in Figure 3.

SECURITY RESULTS. In this section, we give three results about the security of the above anonymous signature scheme. First we consider unforgeability. We prove that if the base signature scheme is existentially unforgeable under chosen message attack and the base commitment scheme has the hiding property, then the anonymous signature scheme associated to them is existentially unforgeable under chosen message attack.

Theorem 4.1 Let $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ be a digital signature scheme and $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ a commitment scheme. Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be the anonymous signature scheme constructed from

```
Alg AKG()                  | Alg ASIG^H(sk, M)              | Alg AVF^H(pk, M, (σ, κ))
(pk, sk) ←$ SKG()          | s ←$ SIG(sk, M)                | (ω, s) ← κ
Return (pk, sk)            | ω ←$ {0,1}^k                   | If (H(ω||s||pk) = σ ∧ |ω| = k) then
                           | σ ←$ H(ω||s||pk)               |     If (SVF(pk, s, M) = 1) then
                           | κ ← (ω, s)                     |         Return 1
                           | Return (σ, κ)                  | Return 0
```

Figure 4: Algorithms used to define the RH construction.

```
Alg AKG()                  | Alg ASIG^H(sk, M)              | Alg AVF^H(pk, M, (σ, κ))
(pk, sk) ←$ SKG()          | s ←$ SIG(sk, M)                | s ← κ
Return (pk, sk)            | σ ←$ H(s||pk)                  | If (H(s||pk) = σ) then
                           | κ ← s                          |     If (SVF(pk, s, M) = 1) then
                           | Return (σ, κ)                  |         Return 1
                           |                                | Return 0
```

Figure 5: Algorithms used to define the DH construction.

$\mathcal{DS}$ and $\mathcal{CMT}$ as in Figure 3. Let F be an adversary against the unforgeability of $\mathcal{AS}$ making $q \geq 1$ queries to oracle $\mathbf{ASign}$. Then there exist adversaries A, B such that

$$\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) \leq 2q \cdot \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A) \; + \; q \cdot \mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(B)\,. \tag{1}$$

Furthermore, the running times of A, B are the same as the running time of F, and A makes q queries to its $\mathbf{Sign}$ oracle.

Due to space limit, the whole proof is deferred to Appendix B.

Next we prove that if the commitment scheme $\mathcal{CMT}$ is hiding then our anonymous signature scheme is anonymous.

Theorem 4.2 Let $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ be a digital signature scheme and $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ a commitment scheme. Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be the anonymous signature scheme constructed from $\mathcal{DS}$ and $\mathcal{CMT}$ as in Figure 3. Let A be an adversary against the anonymity of $\mathcal{AS}$ that makes one query to oracle $\mathbf{CH}$. Then there exists adversary B such that

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) \leq \mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(B)\,. \tag{2}$$

Furthermore, the running time of B is that of A.

Due to space limit, the whole proof is deferred to Appendix C.

Finally, we show that if the commitment scheme is binding then the anonymous signature scheme is unambiguous.

Theorem 4.3 Let $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ be a digital signature scheme and $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ a commitment scheme. Let $\mathcal{AS} = (\mathsf{APG}, \mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be the anonymous signature scheme constructed from $\mathcal{DS}$ and $\mathcal{CMT}$ as in Figure 3. Let A be an adversary against the unambiguity of $\mathcal{AS}$. Then there exists an adversary B such that $\mathbf{Adv}^{\mathrm{unamb}}_{\mathcal{AS}}(A) \leq \mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(B)$. Furthermore, the running time of B is that of A.

Proof: Adversary B runs A to get $(pk_0, pk_1, M_0, M_1, \sigma, \kappa_0, \kappa_1)$. It lets $(s_0, \omega_0) \leftarrow \kappa_0$ and $(s_1, \omega_1) \leftarrow \kappa_1$. Adversary B then outputs $\sigma, (s_0||pk_0, \omega_0), (s_1||pk_1, \omega_1)$.

5 The RH Construction

The Randomized Hash (RH) construction is the result of instantiating the commitment scheme of the CMT construction with the RO-model commitment scheme $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ defined as follows:

```
Alg $\mathsf{CMT}^H(M)$
$\omega \leftarrow_{\$} \{0,1\}^k$
Return $H(\omega || M)$

Alg $\mathsf{CVF}^H(\sigma, M, \omega)$
$\sigma' \leftarrow H(\omega || M)$
If ($|\omega| \neq k$) then return 0
If ($\sigma = \sigma'$) then return 1
```

Figure 4 depicts the algorithms of anonymous signature scheme $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ obtained from the CMT construction of Section 4 applied to a base signature scheme $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ and the commitment scheme we just defined.

We can set the output length k of the RO to 160 bits. (80 bits is not enough because binding reduces to finding collisions and is subject to the birthday attack.) The results of Section 4 imply that the $\mathcal{AS}$ scheme of Figure 4 is secure in the RO model. In this way, we can transform any standard signature scheme into an anonymous one with the following characteristics. The computational overhead is just one hash, meaning signing and verifying are effectively just as efficient as before. The bandwidth overhead is 320 bits: the anonymous signature is 160 bits and the de-anonymizer is 160 bits longer than the base signature. This is pretty good, yet, in what follows, we will provide alternative constructions that reduce the bandwidth overhead even further.

A word of warning. If the base signature scheme already uses a RO (as for instance do FDH [6] and BLS [9]) then the RO H of Figure 4 must be different and independent. This can be ensured by domain seperation as discussed in [5]. This issue arises also below and showed be addressed in the same way.

6 The DH Construction

Base signature schemes such as Schnorr [20], GQ [14] and Fiat-Shamir [13] are randomized, and their signatures have quite a bit of entropy. We will now show that in such cases, the randomizer ω of Figure 4 can be dropped. This saves 160 bits in bandwidth. But the scheme is no longer an instance of the StC transform, and a tailored analysis is needed. We now proceed to detail the construction and provide the analysis.

The DH (Deterministic Hash) construction transforms a base standard signature scheme $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ into an anonymous one $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ using a RO $H : \{0,1\}^* \to \{0,1\}^k$, as shown in Figure 5. For the analysis, we make the following definition.

Definition 6.1 [Min-Entropy of Digital Signatures] Let $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ be a digital signature scheme. The min-entropy $H_\infty(\mathcal{DS})$ of $\mathcal{DS}$ is defined by the equation

$$2^{-H_\infty(\mathcal{DS})} = \max_{(pk,sk),\overline{s},M} \Pr\left[\, \overline{s} = s : s \leftarrow_{\$} \mathsf{SIG}(M, sk) \,\right]$$

where the maximum is over all (pk, sk) that might be output by SKG, all strings $\overline{s}$, and all messages M. ▮

For example, the Schnorr (Sch) scheme [20] over a group of order p has min-entropy $\lg(p)$. A deterministic scheme such as FDH [6] or BLS [9], however, has min-entropy 0. The DH-Sch scheme has bandwidth overhead 160 bits as compared to 320 bits for RH-Sch.

SECURITY. We show that the anonymous signature scheme of Figure 5 is secure in the RO model assuming a secure, high entropy base signature scheme.

Theorem 6.2 Let $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ be a digital signature scheme. Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be the anonymous signature scheme constructed as in Figure 5. Let F be an adversary in the random oracle model against the unforgeability of $\mathcal{AS}$, making q_s queries to oracle $\mathbf{ASign}$, q_H queries to oracle $\mathbf{H}$ and q_O queries to oracle $\mathbf{Open}$. Then there exists adversary A such that

$$\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) \le \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A) + \frac{q_s(q_s + 4(q_H + q_o))}{2^{1+H_\infty(\mathcal{DS})}}\,. \tag{3}$$

Furthermore, the running time of A is that of F and A makes q_O queries to its $\mathbf{Sign}$ oracle.

Theorem 6.3 Let $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ be a digital signature scheme. Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be the anonymous signature scheme constructed as in Figure 5. Let k be the output length of the RO H in the scheme. Let A be an adversary in the random oracle model against the anonymity of $\mathcal{AS}$ making q_H queries to oracle $\mathbf{H}$ and one query to oracle $\mathbf{CH}$. Then

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) \le 2q_H \cdot 2^{-H_\infty(\mathcal{DS})}\,. \tag{4}$$

Theorem 6.4 Let $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ be a digital signature scheme. Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be the anonymous signature scheme constructed as in Figure 5. Let k be the output length of the RO H in the scheme. Let A be an adversary in the random oracle model against the unambiguity of $\mathcal{AS}$ making q_H queries to oracle $\mathbf{H}$. Then we have

$$\mathbf{Adv}^{\mathrm{unamb}}_{\mathcal{AS}}(A) \le \frac{q_H^2}{2^{k+1}}\,. \tag{5}$$

Due to space limit, the whole proofs of the above three theorems are deferred to Appendix D, Appendix E and Appendix F.

7 The Splitting Construction

The splitting construction of anonymous signature is based on the Schnorr protocol [20] and a hash function. The details of Schnorr protocol are deferred to Appendix G. We call it splitting because in our construction, the transcript of the Schnorr protocol is separated into two parts. The message in the first move is viewed as the anonymous signature while the message in the third move is viewed as a de-anonymizer. The associated anonymous signature scheme $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ is defined in Figure 6. Here, and throughout this section, we have fixed a group G of prime order p and a generator g of G. Note that this SP-Sch anonymous signature scheme has zero overhead relative to the base scheme since the full signature is exactly a Schnorr signature. Since the challenge in the Schnorr protocol need be only 80 bits long (not 160) we get an anonymous signature scheme with an 80-bit anonymous signature and a 160 bit de-anonymizer for a 240-bit full signature. Our proof will exploit the general forking lemma of [3], recalled in Appendix H.

SECURITY. First we recall the Discrete Logarithm Assumption that we will use later. Let $G^* = G - \{1\}$ be the set of generators of G, where 1 is the identity element of G. We let $\mathrm{DLog}_g(h)$ denote the discrete logarithm of $h \in G$ to base a generator $g \in G^*$. Let

$$\mathbf{Adv}^{dl}_{G,g}(A) \;=\; \Pr\left[\, x \leftarrow_\$ \mathbb{Z}_p \,;\; x' \leftarrow_\$ A(g, g^x) \;\;:\;\; g^{x'} = g^x \,\right]$$

denote the advantage of an adversary A in attacking the discrete logarithm (dl) problem.

Theorem 7.1 Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be the splitting-based anonymous signature scheme constructed in Figure 6. Let the range of the RO H in the scheme be $\{0,1\}^k \subseteq \mathbb{Z}_p$. Let F be an adversary in

Alg $\mathsf{AKG}()$ $x \leftarrow_{\$} \mathbb{Z}_p$; $X \leftarrow g^x$ Return (X, x)	Alg $\mathsf{ASIG}(sk, M)$ $y \leftarrow_{\$} \mathbb{Z}_p$; $Y \leftarrow g^y$ $x \leftarrow sk$ $\sigma \leftarrow H(X\|\|Y\|\|M)$ $\kappa \leftarrow y + \sigma x \mod p$ Return (σ, κ)	Alg $\mathsf{AVF}(pk, M, (\sigma, \kappa))$ If $X \notin G \vee \|\sigma\| \neq k \vee \kappa \notin \mathbb{Z}_p$ then return 0 $Y \leftarrow g^{\kappa} \cdot X^{-\sigma}$ If $\sigma = H(X\|\|Y\|\|M)$ then return 1 Else return 0

Figure 6: Algorithms used to define the splitting-based anonymous signature scheme. Here G is a group of prime order p and g is a generator of G.

the random oracle model against the unforgeability of $\mathcal{AS}$, making q_s queries to oracle **ASign**, q_H queries to oracle **H** and having running time at most t_F. Then there exists an algorithm B that attacks the discrete logarithm problem with advantage frk such that

$$\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) \leq \frac{q_s^2 + 4q_s q_H + 2q_s q_o}{2p} + \frac{q_H}{p} + \sqrt{q_H \cdot \mathrm{frk}}$$

Furthermore, the running time of B is $2t_F$.

Theorem 7.2 Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be the splitting-based anonymous signature scheme constructed in Figure 6. Let the range of the RO H in the scheme be $\{0,1\}^k \subseteq \mathbb{Z}_p$. Let A be an adversary in the random oracle model against the anonymity of $\mathcal{AS}$ making q_H queries to oracle **H** and one query to oracle **LR**. Then

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) \leq 2q_H/p\,. \tag{6}$$

Theorem 7.3 Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be the splitting-based anonymous signature scheme constructed in Figure 6. Let the range of the RO H in the scheme be $\{0,1\}^k \subseteq \mathbb{Z}_p$. Let A be an adversary in the random oracle model against the unambiguity of $\mathcal{AS}$ making q_H queries to oracle **H**. Then

$$\mathbf{Adv}^{\mathrm{unamb}}_{\mathcal{AS}}(A) \leq q_H^2/2^{k+1}\,.$$

Due to space limit, the whole proofs of the above theorems are deferred to Appendix H, Appendix I and Appendix J respectively.

8 A Reverse Connection

From the primitive definitions, we can see that anonymous signatures (AS) and commitment schemes (CMT) share something in common. Firstly, AS hide the identity of the signer while CMT hide the committed message. Secondly, in the AS setting the signature can not be opened under a different public key while in the CMT setting the committed message can not be opened in a different way. Do these imply that when we have a scheme of one primitive we can transform it to that of the other primitive? We have showed one direction in our CMT construction in Section 4. To complete the whole picture, we are going to propose a generic transformation, to convert any anonymous signature scheme into a commitment scheme. However the similarities between these two primitives don't imply that it is trivial to find such a transformation,

```
Alg $\mathsf{CMT}(M)$
$(pk_0, sk_0) \leftarrow\$ \mathsf{AKG}()$
$(pk_1, sk_1) \leftarrow\$ \mathsf{AKG}()$
If $(pk_0 = pk_1)$ then $bad \leftarrow \texttt{true}$
$n \leftarrow |M|$
For $i = 1$ to $n$
    $(\sigma_i, \kappa_i) \leftarrow\$ \mathsf{ASIG}(sk_{M[i]}, i)$
$\sigma \leftarrow (0, \sigma_1||\ldots||\sigma_n||pk_0||pk_1)$
$\omega \leftarrow \kappa_1||\ldots||\kappa_n$
If $bad = \texttt{true}$ then $\sigma \leftarrow (1, M)$ ; $\omega \leftarrow M$
Return $(\sigma, \omega)$
```

```
Alg $\mathsf{CVF}(\sigma, M, \omega)$
$(b, \sigma') \leftarrow \sigma$
If $(b = 1)$ then
    If $(\sigma' = M \wedge \omega = M)$ then return 1
    Else return 0
Else
    $\kappa_1||\ldots||\kappa_n \leftarrow \omega$
    $\sigma_1||\ldots||\sigma_n||pk_0||pk_1 \leftarrow \sigma'$
    If $(pk_0 = pk_1)$ then return 0
    For $i = 1$ to $n$ $d_i \leftarrow \mathsf{AVF}(pk_{M[i]}, i, (\sigma_i, \kappa_i))$
    Return $d_1 \wedge \ldots \wedge d_n$
```

Figure 7: CMT construction from AS.

especially an efficient one. Our transformation, which provides a direct and efficient conversion from AS to CMT, is depicted in Figure 7.

SECURITY OF OUR CONSTRUCTION. We prove that if the given anonymous signature scheme can achieve unforgeability, anonymity and unambiguity, then the commitment scheme obtained using our construction has the property of hiding and binding. For the analysis, we use the following game to capture the situation that two independently generated public keys are the same. And we use Lemma 8.1 to bound the probability that such public key collision happens.

```
procedure Initialize // $\mathrm{PKColl}_{\mathcal{AS}}$
$(pk_0, sk_0) \leftarrow\$ \mathsf{AKG}()$
$(pk_1, sk_1) \leftarrow\$ \mathsf{AKG}()$
Return $(pk_0 = pk_1)$
```

Lemma 8.1 Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be an anonymous signature scheme. Then there is an adversary F against the unforgeability of $\mathcal{AS}$ such that $\Pr[\mathrm{PKColl}_{\mathcal{AS}}] \leq \mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F)$. The running time of F is that of AKG and F makes no oracle queries.

Proof: On input pk F let $pk_0 \leftarrow pk$ and $(pk_1, sk_1) \leftarrow\$ \mathsf{AKG}$. It let M be any message, for example $M = 0$. It lets $(\sigma, \kappa) \leftarrow\$ \mathsf{ASIG}(sk, M)$ and returns $(M, (\sigma, \kappa))$. If $pk_1 = pk_0$, then it wins the game AUF-CMA$_{\mathcal{AS}}$, so we have $\Pr[\mathrm{PKColl}_{\mathcal{AS}}] \leq \mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F)$.

Theorem 8.2 Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be an anonymous signature scheme and $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ the commitment scheme constructed from $\mathcal{AS}$ as in Figure 7. Let A be an adversary against the hiding property of $\mathcal{CMT}$, making one query to oracle $\mathbf{LR}$, this always consisting of a pair of n-bit messages, and having running time at most t_A. Then there exists adversary B making n queries to oracle $\mathbf{CH}$ and adversary F making no queries such that

$$\mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(A) \leq n \cdot \mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(B) + 2 \cdot \mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) .$$

Furthermore, the running times of B and F are the same as that of A. B makes one query to its $\mathbf{CH}$ oracle and F makes no queries.

Due to space limit, the whole proof is deferred to Appendix K.

Theorem 8.3 Let $\mathcal{AS} = (\mathsf{AKG}, \mathsf{ASIG}, \mathsf{AVF})$ be an anonymous signature scheme and $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ the commitment scheme constructed from $\mathcal{AS}$ as in Figure 7. Let A be an adversary against the binding property of $\mathcal{CMT}$. Then there exists an adversary B such that

$$\mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(A) \leq \mathbf{Adv}^{\mathrm{unamb}}_{\mathcal{AS}}(B) \ .$$

Furthermore, the running time of B is that of A.

Due to space limit, the whole proof is deferred to Appendix L.

References

[1] M. Abdalla, J. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. *EUROCRYPT 2002*, LNCS 2332, Springer-Verlag.

[2] K. Barr and K. Asanovic. Energy aware lossless data compression. *MobiSys 2003*, ACM Press.

[3] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. *ACM CCS 2006*, ACM Press.

[4] M. Bellare, D. Micciancio and B. Warinschi. Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. *EUROCRYPT 2003*, LNCS 2656, Springer-Verlag.

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *ACM CCS 1993*, ACM Press.

[6] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. *EUROCRYPT 1996*, LNCS 1070, Springer-Verlag.

[7] M. Bellare and P. Rogaway. The Security of Triple Encryption and a Framwork for Code-Based Game-Playing Proofs. *EUROCRYPT 2006*, LNCS 4004, Springer-Verlag.

[8] A. Bender, J. Katz, and R. Morselli. Ring signatures: stronger definitions, and constructions without random oracles. *TCC 2006*, LNCS 3876, Springer-Verlag.

[9] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297-319.

[10] D. Chaum and E. Heyst. Group signatures. *EUROCRYPT 1991*, LNCS 547, Springer-Verlag.

[11] C. Dwork, M. Naor, O. Reingold and L. Stockmeyer. Magic functions. *Journal of the ACM 2003*, 50(6):852-921.

[12] M. Fischlin. Anonymous signatures made easy. *PKC 2007*, LNCS 4450, Springer-Verlag.

[13] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *CRYPTO 1986*, LNCS 263, Springer-Verlag.

[14] L. Guillou and J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. *CRYPTO 1988*, LNCS 403, Springer-Verlag.

[15] J. Håstad, R. Impagliazzo, L. Levin and M. Luby. A Pseudorandom generator from any one-way function. *SIAM Journal on Computing 1999*, 28(4):1364–1396.

[16] M. Noar. Bit Commitment Using Pseudorandomness. *Journal of Cryptology 1991*, 4:151-158.

[17] N. Nisan and A. Ta-Shma. Extracting randomness: a survey and new constructions. *Journal of Computer and System Sciences*, 58(1):149-173.

[18] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43-52.

[19] R. Rivest, A. Shamir and Y. Tauman. How to leak a secret. *ASIACRYPT 2001*, LNCS 2248, Springer-Verlag.

```
Initialize
$b \leftarrow\!\$ \{0,1\}$

LR($M_0, M_1$)
If ($|M_0| \neq |M_1|$) then return $\perp$
$(\sigma, \omega) \leftarrow\!\$ \mathsf{CMT}(M_b)$
Return $\sigma$

Finalize($d$)
Return $(b = d)$
```

```
Initialize

Finalize($\sigma, (M_0, \omega_0), (M_1, \omega_1)$)
$d_0 \leftarrow (\mathsf{CVF}(\sigma, M_0, \omega_0) = 1)$
$d_1 \leftarrow (\mathsf{CVF}(\sigma, M_1, \omega_1) = 1)$
Return $(d_0 \wedge d_1 \wedge M_0 \neq M_1)$
```

```
Initialize
$(pk, sk) \leftarrow\!\$ \mathsf{SKG}()$ ; $i \leftarrow 0$ ; $S \leftarrow \emptyset$
Return $pk$

Sign($M$)
$i \leftarrow i + 1$ ; $M_i \leftarrow M$
$S \leftarrow S \cup \{M_i\}$ ; $s_i \leftarrow\!\$ \mathsf{SIG}(sk, M)$
Return $\sigma_i$

Finalize($M, s$)
Return $(M \notin S \wedge \mathsf{SVF}(pk, s, M) = 1)$
```

Figure 8: Game HIDE in the left used to define hiding and game BIND in the center used to define binding of commitment scheme $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$. Game EUF-CMA in the right used to define existential unforgeability of signature scheme $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$.

[20] C. SCHNORR. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[21] G. YANG, D. WONG, X. DENG, AND H. WANG. Anonymous signature schemes. *PKC 2006*, LNCS 3958, Springer-Verlag.

A Security Definitions of Signatures and Commitments

The advantage of an adversary F in attacking the unforgeability is

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(F) = \Pr\left[\,\text{EUF-CMA}^{F}_{\mathcal{DS}}\,\right] ,$$

where game EUF-CMA is shown in Figure 8.
The advantage of an adversary A in attacking the hiding property is

$$\mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(A) = 2 \cdot \Pr\left[\,\text{HIDE}^{A}_{\mathcal{CMT}}\,\right] - 1 .$$

where game HIDE is in Figure 8. In the game, A is allowed only one query to its **LR** oracle. The advantage of an adversary A in attacking the binding property is

$$\mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(A) = \Pr\left[\,\text{BIND}^{A}_{\mathcal{CMT}}\,\right]$$

where game BIND is in Figure 8.

B Proof of Theorem 4.1

Proof: We use games G_0, G_1, G_2, G_3, G_4 of Figure 9, where l denotes the length of a signature in $\mathcal{DS}$. We assume wlog that F always makes exactly q queries to **ASign** rather than at most q. Note that G_0 and G_1 are different only in procedure **Finalize**. For G_0, any execution with F in which the outcome is `true` satisfies $M \notin S$. For G_1, any execution with F in which the outcome is `true` satisfies $M \in S$. So we have

$$\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) \;\leq\; \Pr\left[\,G_0^F\,\right] + \Pr\left[\,G_1^F\,\right] . \tag{7}$$

Games G_1 and G_2 are identical except for the first condition in the procedure **Finalize**. Any execution of G_2 with F in which the outcome is `true` must have not only $M \in S$ but also $M = M_g$. On the other hand G_1 does not use g anywhere and thus the events G_1^F and $M = M_g$ are independent and the probability of the latter is $1/q$. Hence, we have

$$\Pr\left[\,G_1^F\,\right] \leq q \cdot \Pr\left[\,G_2^F\,\right] . \tag{8}$$

```
Initialize // $G_0$, $G_1$,$G_2$, $G_3$, $G_4$
    $(pk, sk) \leftarrow\$ \mathsf{SKG}()$
    $S \leftarrow \emptyset$ ; $E \leftarrow \emptyset$ ; $i \leftarrow 0$ ; $j \leftarrow 0$
    $g \leftarrow\$ \{1, \ldots, q\}$
    Return $pk$

Open($j$) // $G_0$, $G_1$,$G_2$, $\boxed{G_3}$, $\boxed{G_4}$
    If ($j \leq 0 \vee j > i$) Return $\perp$
    $E \leftarrow E \cup \{M_j\}$
    If ( $j = g$ ) then $bad \leftarrow$ true; $\boxed{\kappa_j \leftarrow \perp}$
    Return $\kappa_j$

ASign($M$) // $G_0$,$G_1$,$G_2$,$G_3$
    $i \leftarrow i + 1$ ; $M_i \leftarrow M$ ; $S \leftarrow S \cup \{M_i\}$
    $s_i \leftarrow\$ \mathsf{SIG}(sk, M)$
    $(\sigma_i, \omega_i) \leftarrow\$ \mathsf{CMT}(s_i||pk)$ ; $\kappa_i \leftarrow (s_i, \omega_i)$
    Return $\sigma_i$

ASign($M$) // $G_4$
    $i \leftarrow i + 1$ ; $M_i \leftarrow M$ ; $S \leftarrow S \cup \{M_i\}$
    If ( $i = g$ ) then $s_i \leftarrow\$ \{0,1\}^l$
    else $s_i \leftarrow\$ \mathsf{SIG}(sk, M_i)$
    $(\sigma_i, \omega_i) \leftarrow\$ \mathsf{CMT}(s_i||pk)$
    $\kappa_i \leftarrow (s_i, \omega_i)$
    Return $\sigma_i$

Finalize($M, (\sigma, \kappa)$) // $G_0$
    Return $(M \notin S \wedge M \notin E \wedge \mathsf{AVF}(pk, M, (\sigma, \kappa)) = 1)$

Finalize($M, (\sigma, \kappa)$) // $G_1$
    Return $(M \in S \wedge M \notin E \wedge \mathsf{AVF}(pk, M, (\sigma, \kappa)) = 1)$

Finalize($M, (\sigma, \kappa)$) // $G_2$, $G_3$, $G_4$
    Return $(M = M_g \wedge M \notin E \wedge \mathsf{AVF}(pk, M, (\sigma, \kappa)) = 1)$
```

Figure 9: Game sequence used in proof of Theorem 4.1. Game G_3, G_4 include the boxed code while G_0, G_1, G_2 do not.

The difference between G_3 and G_2 is that the former includes the boxed code in **Open**. But any execution of G_3 with F in which the outcome is `true` must have $M = M_g$ and $M \notin E$, so the boxed code would not have been executed. Recall that BD_i denotes the event that bad is set to `true` in game G_i. Then based on Lemma 2.1, we have

$$\Pr\left[\, G_2^F \,\right] \;=\; \Pr\left[\, G_2^F \wedge \overline{\mathsf{BD}}_2 \,\right] \;=\; \Pr\left[\, G_3^F \wedge \overline{\mathsf{BD}}_3 \,\right] \,. \tag{9}$$

Combining (7), (8) and (9), we get

$$\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS},F}(k) \;\leq\; \Pr\left[\, G_0^F \,\right] \;+\; q \cdot \Pr\left[\, G_3^F \wedge \overline{\mathsf{BD}}_3 \,\right] \,. \tag{10}$$

We will build A_0, A_1, B so that

$$\Pr\left[\, G_0^F \,\right] \;\leq\; \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A_0) \tag{11}$$

$$\Pr\left[\, G_3^F \wedge \overline{\mathsf{BD}}_3 \,\right] - \Pr\left[\, G_4^F \wedge \overline{\mathsf{BD}}_4 \,\right] \;\leq\; \mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(B) \tag{12}$$

$$\Pr\left[\, G_4^F \wedge \overline{\mathsf{BD}}_4 \,\right] \;\leq\; \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A_1) \tag{13}$$

A_0, A_1 will make q oracle queries and A_0, A_1, B will have the same running time as F. Now let A on input pk pick $c \leftarrow\$ \{0,1\}$ and run $A_c(pk)$. Then

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A) \;=\; \frac{1}{2}\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A_0) + \frac{1}{2}\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A_1) \,. \tag{14}$$

Equation (1) follows from (10), (11) (12), (13) and (14). We proceed to describe A_0, A_1, B.

Adversary A_0 gets input pk and then does the following initializations:

$$S \leftarrow \emptyset\,;\; E \leftarrow \emptyset\,;\; i \leftarrow 0\,;\; j \leftarrow 0\,;\; g \leftarrow\$ \{1, \ldots, q\} \,. \tag{15}$$

It then runs $F(pk)$. It answers F's queries to **ASign** using the following procedure:

procedure **ASign**(M)

```
$i \leftarrow i+1$ ; $M_i \leftarrow M$ ; $S \leftarrow S \cup \{M_i\}$
$s_i \leftarrow\!\!\$\ \mathbf{Sign}(M)$
$(\sigma_i, \omega_i) \leftarrow\!\!\$\ \mathsf{CMT}(s_i||pk)$ ; $\kappa_i \leftarrow (s_i, \omega_i)$
Return $\sigma_i$
```

A_0 answers F's queris to $\mathbf{Open}$ exactly as G_0 does. Finally, F outputs $(M, (\sigma, \kappa))$. Adversary A_0 parses κ to (s, ω) and then outputs (M, s).

Adversary B against the hiding property of $\mathcal{CMT}$ begins by executing the code of the $\mathbf{Initialize}$ procedure of G_3, thereby defining for itself the parameters pk, sk, S, E, i, j, g. It then starts running F on pk. It answers F's queries to $\mathbf{ASign}$ using the following procedure:

```
procedure $\mathbf{ASign}(M)$
$i \leftarrow i+1$ ; $M_i \leftarrow M$ ; $S \leftarrow S \cup \{M_i\}$
$s_i \leftarrow\!\!\$\ \mathsf{SIG}(sk, M)$
If ( $i = g$ ) then $s_0 \leftarrow\!\!\$\ \{0,1\}^l$ ; $\sigma_i \leftarrow \mathbf{LR}(s_0||pk, s_i||pk)$
else $(\sigma_i, \omega_i) \leftarrow\!\!\$\ \mathsf{CMT}(s_i||pk)$ ; $\kappa_i \leftarrow (s_i, \omega_i)$
Return $\sigma_i$
```

It answers F's queries to $\mathbf{Open}$ exactly as G_3 does. Finally, F outputs $(M, (\sigma, \kappa))$. Adversary B outputs 1 if $M = M_g \wedge M \notin E \wedge \mathsf{AVF}(pk, M, (\sigma, \kappa)) = 1$, and 0 otherwise. Letting d denote the output of B, we have

$$\Pr[\, d = 1 \mid b = 1 \,] = \Pr\left[\, G_3^F \wedge \overline{\mathsf{BD}}_3 \,\right]$$
$$\Pr[\, d = 1 \mid b = 0 \,] = \Pr\left[\, G_4^F \wedge \overline{\mathsf{BD}}_4 \,\right]$$

in game $\mathrm{HIDE}_{\mathcal{CMT}}^{B_2}$. Subtracting, we get

$$\Pr\left[\, G_3^F \wedge \overline{\mathsf{BD}}_3 \,\right] - \Pr\left[\, G_4^F \wedge \overline{\mathsf{BD}}_4 \,\right] = \mathbf{Adv}_{\mathcal{CMT}}^{\mathrm{hide}}(B) \ .$$

Adversary A_1 gets input pk and then does the initializations (15). It then runs $F(pk)$. It answers F's queries to $\mathbf{ASign}$ using the following procedure:

```
procedure $\mathbf{ASign}(M)$
$i \leftarrow i+1$ ; $M_i \leftarrow M$ ; $S \leftarrow S \cup \{M_i\}$
If ( $i = g$ ) then $s_i \leftarrow\!\!\$\ \{0,1\}^l$
Else $s_i \leftarrow\!\!\$\ \mathbf{Sign}(M)$
$(\sigma_i, \omega_i) \leftarrow\!\!\$\ \mathsf{CMT}(s_i||pk)$ ; $\kappa_i \leftarrow (s_i, \omega_i)$
Return $\sigma_i$
```

It answers F's queris to $\mathbf{Open}$ exactly as G_4 does. Finally, F outputs $(M, (\sigma, \kappa))$. A_1 parses κ to (s, ω) and outputs (M, s).

C Proof of Theorem 4.2

Proof: Adversary B begins with $(pk_i, sk_i) \leftarrow\!\!\$\ \mathsf{AKG}()$ for $i = 0, 1$. It then runs $A((pk_0, sk_0), (pk_1, sk_1))$ and answers A's queries to $\mathbf{CH}$ using the following procedure:

```
**Initialize** // $G_0 - G_6$
$(pk, sk) \leftarrow\$ \mathsf{SKG}()$
$E \leftarrow \emptyset$ ; $U \leftarrow \emptyset$ ; $i \leftarrow 0$
Return $pk$

**ASign**$(M)$ // $\boxed{G_0}$, $G_1$
$i \leftarrow i + 1$ ; $M_i \leftarrow M$
$s_i \leftarrow\$ \mathsf{SIG}(sk, M_i)$ ; $\sigma_i \leftarrow\$ \{0,1\}^k$
$S \leftarrow \{j : 1 \le j < i \wedge s_j = s_i\}$
If $S \neq \emptyset$ then $j \leftarrow\$ S$ ; $\sigma_i \leftarrow \sigma_j$
Else if $H[s_i||pk]$ then $\sigma_i \leftarrow H[s_i||pk]$
$\boxed{H[s_i||pk] \leftarrow \sigma_i}$
Return $\sigma_i$

**ASign**$(M)$ // $\boxed{G_2}$, $G_3$
$i \leftarrow i + 1$ ; $M_i \leftarrow M$
$s_i \leftarrow\$ \mathsf{SIG}(sk, M_i)$ ; $\sigma_i \leftarrow\$ \{0,1\}^k$
$S \leftarrow \{j : 1 \le j < i \wedge s_j = s_i\}$
If $S \neq \emptyset$ then bad $\leftarrow$ true; $\boxed{j \leftarrow\$ S \text{ ; } \sigma_i \leftarrow \sigma_j}$
Else if $H[s_i||pk]$ then bad $\leftarrow$ true; $\boxed{\sigma_i \leftarrow H[s_i||pk]}$
Return $\sigma_i$

**ASign**$(M)$ // $G_4, G_5$
$i \leftarrow i + 1$ ; $M_i \leftarrow M$ ; $s_i \leftarrow\$ \mathsf{SIG}(sk, M_i)$ ; $\sigma_i \leftarrow\$ \{0,1\}^k$
Return $\sigma_i$

**ASign**$(M)$ // $G_6$
$i \leftarrow i + 1$ ; $M_i \leftarrow M$ ; $\sigma_i \leftarrow\$ \{0,1\}^k$
Return $\sigma_i$

**Finalize**$(M, (\sigma, \kappa))$ // $G_0 - G_6$
Return $(M \notin E \wedge H[s||pk] = \sigma \wedge \mathsf{SVF}(pk, s, M) = 1)$

**Open**$(j)$ // $G_0$, $\boxed{G_1}$, $\boxed{G_2}$, $\boxed{G_3}$, $\boxed{G_4}$, $\boxed{G_5}$
If $(j \le 0 \vee j > i)$ Return $\perp$
$E \leftarrow E \cup \{M_j\}$ ; $U \leftarrow U \cup \{j\}$
$\boxed{H[s_j||pk] \leftarrow \sigma_j}$
Return $s_j$

**Open**$(j)$ // $G_6$
If $(j \le 0 \vee j > i)$ Return $\perp$
$s_j \leftarrow\$ \mathsf{SIG}(sk, M_j)$ ; $E \leftarrow E \cup \{M_j\}$ ; $U \leftarrow U \cup \{j\}$
$H[s_j||pk] \leftarrow \sigma_j$
Return $s_j$

**H**$(x)$ // $G_1, G_2, G_3$
If $(H[x])$ Return $H[x]$
$s||pk \leftarrow x$ ; $H[x] \leftarrow\$ \{0,1\}^k$
$T \leftarrow \{j : 1 \le j \le i \wedge s = s_j \wedge j \notin U\}$
If $(T \neq \emptyset)$ then $j \leftarrow\$ T$ ; $H[x] \leftarrow \sigma_j$
Return $H[x]$

**H**$(x)$ // $\boxed{G_4}$, $G_5$
If $(H[x])$ Return $H[x]$
$s||pk \leftarrow x$ ; $H[x] \leftarrow\$ \{0,1\}^k$
$T \leftarrow \{j : 1 \le j \le i \wedge s = s_j \wedge j \notin U\}$
If $(T \neq \emptyset)$ then bad $\leftarrow$ true; $\boxed{j \leftarrow\$ T \text{ ; } H[x] \leftarrow \sigma_j}$
Return $H[x]$

**H**$(x)$ // $G_0, G_6$
If $(H[x])$ Return $H[x]$
$H[x] \leftarrow\$ \{0,1\}^k$
Return $H[x]$
```

Figure 10: Game sequence used in proof of Theorem 6.2.

```
**procedure CH**$(M)$
$s_0 \leftarrow\$ \mathsf{SIG}(sk_0, M)$ ; $s_1 \leftarrow\$ \mathsf{SIG}(sk_1, M)$
$\sigma \leftarrow \mathbf{LR}(s_0||pk_0, s_1||pk_1)$
Return $\sigma$
```

After A outputs its guess d, adversary B outputs the same d. We have

$$\begin{aligned}\Pr\left[\,\mathrm{HIDE}^B_{\mathcal{CMT}} \mid b = 1\,\right] &= \Pr\left[\,\mathrm{ANON}^A_{\mathcal{AS}} \mid b = 1\,\right]\\ \Pr\left[\,\mathrm{HIDE}^B_{\mathcal{CMT}} \mid b = 0\,\right] &= \Pr\left[\,\mathrm{ANON}^A_{\mathcal{AS}} \mid b = 0\,\right]\end{aligned}$$

in game $\mathrm{HIDE}^B_{\mathcal{CMT}}$, from which (2) follows.

D Proof of Theorem 6.2

Proof: We refer to the games of Figure 10. Game G_0 is equivalent to $\mathrm{AUF\text{-}CMA}_{\mathcal{AS}}$, so

$$\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) = \Pr\left[\,G_0^F\,\right] .$$

Game G_1 omits the boxed code in $\mathbf{ASign}$, meaning $H[s_i||pk]$ is not assigned σ_i at this point. Instead the assignment is delayed, being done by $H(x)$ or $\mathbf{Open}$ as necessary . So

$$\Pr\left[\,G_0^F\,\right] = \Pr\left[\,G_1^F\,\right] .$$

But G_1, G_2 are equivalent and G_2 and G_3 are identical until bad, so by Lemma 2.1

$$\begin{aligned}\Pr\left[\,G_1^F\,\right] &= \Pr\left[\,G_2^F\,\right]\\ &= \Pr\left[\,G_3^F\,\right] + \Pr\left[\,G_2^F\,\right] - \Pr\left[\,G_3^F\,\right]\\ &\le \Pr\left[\,G_3^F\,\right] + \Pr\left[\,\mathsf{BD}_3\,\right]\end{aligned}$$

G_3 and G_4 are equivalent and G_4 and G_5 are identical until bad, so by Lemma 2.1

$$\begin{aligned}\Pr\left[\,G_3^F\,\right] &= \Pr\left[\,G_4^F\,\right]\\ &= \Pr\left[\,G_5^F\,\right] + \Pr\left[\,G_4^F\,\right] - \Pr\left[\,G_5^F\,\right]\\ &\le \Pr\left[\,G_5^F\,\right] + \Pr\left[\,\mathsf{BD}_5\,\right]\end{aligned}$$

In G_5, the signature s_i for $i \notin U$ is unused beyond for setting *bad*, so in G_6 we don't compute it. We have

$$\Pr\left[\,G_5^F\,\right] = \Pr\left[\,G_6^F\,\right] .$$

Putting the above together we have

$$\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) \le \Pr\left[\,G_6^F\,\right] + \Pr\left[\,\mathsf{BD}_3\,\right] + \Pr\left[\,\mathsf{BD}_5\,\right] . \tag{16}$$

Adversary A sets input pk and perform the initialization $E \leftarrow \emptyset$; $U \leftarrow \emptyset$; $i \leftarrow 0$. It then runs $F(pk)$. It responds to $\mathbf{H}$ and $\mathbf{ASign}$ queries as does G_6, and to $\mathbf{Open}$ queries via the $\mathbf{Open}$ procedure of G_6 except that the computation $\mathsf{SIG}(sk, M_i)$ is substituted by a call $\mathbf{Sign}(M_i)$ to A's sign oracle. A outputs the same thing as F. We have

$$\Pr\left[\,G_6^F\,\right] \le \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A) \tag{17}$$

Now

$$\begin{aligned}\Pr\left[\,\mathsf{BD}_3\,\right] &\le \sum_{i=1}^{q_s}\left(\frac{i-1}{2^{H_\infty(\mathcal{DS})}} + \frac{q_H + q_o}{2^{H_\infty(\mathcal{DS})}}\right)\\ &= \frac{q_s(q_s-1) + 2q_s(q_H+q_o)}{2^{1+H_\infty(\mathcal{DS})}}\end{aligned} \tag{18}$$

Finally the maximum size of T in procedure H of G_5 is q_s and hence

$$\Pr\left[\,\mathsf{BD}_5\,\right] \le \frac{q_s q_H}{2^{H_\infty(\mathcal{DS})}} . \tag{19}$$

Putting together (16), (17), (18) and (19) completes the proof.

E Proof of Theorem 6.3

Proof: We use games G_0, G_1, G_2 of Figure 11. So we have

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) = 2 \cdot \Pr\left[\,G_0^A\,\right] - 1 . \tag{20}$$

```
Initialize // $G_0, G_1, G_2$
    $b \leftarrow_{\$} \{0,1\}$
    $(pk_0, sk_0) \leftarrow_{\$} \mathsf{SKG}()$
    $(pk_1, sk_1) \leftarrow_{\$} \mathsf{SKG}()$
    Return $((pk_0, sk_0), (pk_1, sk_1))$

CH($M$) // $G_0$
    $s \leftarrow_{\$} \mathsf{SIG}(sk_b, M)$ ; $\sigma \leftarrow \mathbf{H}(s||pk_b)$
    Return $\sigma$

Finalize($d$) // $G_0, G_1, G_2$
    Return $(b = d)$

CH($M$) // $\boxed{G_1}$,$G_2$
    $s \leftarrow_{\$} \mathsf{SIG}(sk_b, M)$ ; $\sigma \leftarrow_{\$} \{0,1\}^k$
    If ($H[s||pk_b]$) then bad $\leftarrow$ true ; $\boxed{\sigma \leftarrow H[s||pk_b]}$
    $H[s||pk_b] \leftarrow \sigma$
    Return $\sigma$

H($x$) // $G_0, G_1, G_2$
    If ($H[x]$) Return $H[x]$
    $H[x] \leftarrow_{\$} \{0,1\}^k$
    Return $H[x]$
```

Figure 11: Game sequence used in proof of Theorem 6.3.

Games G_0 and G_1 are equivalent, and G_1 and G_2 are identical until *bad* so by Lemma 2.1 we have

$$\begin{aligned}\Pr\left[\,G_0^A\,\right] &= \Pr\left[\,G_1^A\,\right]\\ &= \Pr\left[\,G_1^A\,\right] - \Pr\left[\,G_2^A\,\right] + \Pr\left[\,G_2^A\,\right]\\ &\leq \Pr\left[\,\mathsf{BD}_2\,\right] + \Pr\left[\,G_2^A\,\right] \qquad (21)\end{aligned}$$

Combining (20) and (21), we get

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) \leq 2 \cdot (\Pr\left[\,G_2^A\,\right] + \Pr\left[\,\mathsf{BD}_2\,\right]) - 1\ .$$

In game G_2, the challenge signature $H[s||pk_b]$ is set to be a random string with length k, so we have $\Pr\left[\,G_2^A\,\right] = 1/2$ and thus

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) \leq 2 \cdot \Pr\left[\,\mathsf{BD}_2\,\right]\ . \qquad (22)$$

In game G_2, *bad* is set `true` when the signature generated in $\mathbf{LR}$ is equal to some x which A queried to $\mathbf{H}$, so we have

$$\Pr\left[\,\mathsf{BD}_2\,\right] \leq q_H \cdot 2^{-H_\infty(\mathcal{DS})}\ . \qquad (23)$$

Equation (4) follows from (22) and (23).

F Proof of Theorem 6.4

Proof: Let $(pk_0, pk_1, M_0, M_1, \sigma, \kappa_0, \kappa_1)$ denote the output of A. Let $s_0 \leftarrow \kappa_0$ and $s_1 \leftarrow \kappa_1$. If A wins the game $\mathrm{UNAMB}_{\mathcal{AS}}$, then we have $H(s_0||pk_0) = H(s_1||pk_1) = \sigma$ but $pk_0 \neq pk_1$, meaning that we have a collision for H. Since A makes q_H queries to H we have (5).

We remark that, as the proof shows, for unambiguity. we need only that H is collision resistant rather than a RO.

G Schnorr Identification Protocol

In Figure 12, we give the description of Schnorr Identification protocol, on which our splitting construction in Figure 6 is based.

Algorithm KG	**Prover**		**Verifier**
$x \leftarrow_{\$} \mathbb{Z}_p$	Input: $sk = x$		Input: $pk = X$
$X \leftarrow g^x$	$y \leftarrow_{\$} \mathbb{Z}_p$		
$pk \leftarrow X$	$Y \leftarrow g^y$	$\xrightarrow{Y}$	
$sk \leftarrow x$		$\xleftarrow{\sigma}$	
Return (pk, sk)	$\kappa \leftarrow y + \sigma x \mod p$	$\xrightarrow{\kappa}$	If $g^\kappa = YX^\sigma$ then $\textsc{Dec} \leftarrow 1$ else $\textsc{Dec} \leftarrow 0$
			Return $\textsc{Dec}$

Figure 12: Schnorr Identification Protocol used in Section 7.

H Proof of Theorem 7.1

Before giving the security proof, we first recall the general forking lemma [3], which will be used later.

Lemma H.1 [General Forking Lemma] Fix an integer $q \geq 1$ and a set H of size $h \geq 2$. Let A be a randomized algorithm that on input $X, h_1, \ldots, h_q$ returns a pair, the first element of which is an integer in the range $0, \ldots, q$ and the second element of which we refer to as a *side output*. Let *IG* be a randomized algorithm that we call the input generator. The *accepting probability* of A, denoted acc, is defined as the probability that $J \geq 1$ in the experiment

$$X \leftarrow_{\$} \mathit{IG} \; ; \; h_1, \ldots, h_q \leftarrow_{\$} H \; ; \; (J, s) \leftarrow_{\$} A(X, h_1, \ldots, h_q) \; .$$

The *forking algorithm* F_A associated to A is the randomized algorithm that on input x proceeds as follows:

```
Algorithm $F_A(x)$
    Pick coins $\rho$ for $A$ at random
    $h_1, \ldots, h_q \leftarrow_{\$} H$
    $(I, s) \leftarrow A(x, h_1, \ldots, h_q; \rho)$
    If $I = 0$ then return $(0, \varepsilon, \varepsilon)$
    $h'_I, \ldots, h'_q \leftarrow_{\$} H$
    $(I', s') \leftarrow A(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q; \rho)$
    If ($I = I'$ and $h_I \neq h'_I$) then return $(1, s, s')$
    Else return $(0, \varepsilon, \varepsilon)$.
```

Let

$$\mathrm{frk} \;=\; \Pr\left[\, b = 1 \;\;:\;\; X \leftarrow_{\$} \mathit{IG} \; ; \; (b, s, s') \leftarrow_{\$} F_A(X) \,\right] \; .$$

Then

$$\mathrm{frk} \;\geq\; \mathrm{acc} \cdot \left(\frac{\mathrm{acc}}{q} - \frac{1}{h} \right) \; . \tag{24}$$

Alternatively,

$$\mathrm{acc} \;\leq\; \frac{q}{h} + \sqrt{q \cdot \mathrm{frk}} \; . \tag{25}$$

Proof: Let $q = q_s + q_H$ and consider games $G_0 - G_7$ of Figure 13. We have

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) &= \Pr\left[\, G_0^F \,\right] \\
&= \Pr\left[\, G_1^F \,\right] \\
&= \Pr\left[\, G_2^F \,\right] \\
&= \Pr\left[\, G_3^F \,\right] + \Pr\left[\, G_2^F \,\right] - \Pr\left[\, G_3^F \,\right] \\
&\le \Pr\left[\, G_3^F \,\right] + \Pr\left[\, \mathsf{BD}_3 \,\right] .
\end{aligned}$$

And

$$\begin{aligned}
\Pr\left[\, G_3^F \,\right] &= \Pr\left[\, G_4^F \,\right] \\
&= \Pr\left[\, G_5^F \,\right] + \Pr\left[\, G_4^F \,\right] - \Pr\left[\, G_5^F \,\right] \\
&\le \Pr\left[\, G_5^F \,\right] + \Pr\left[\, \mathsf{BD}_5 \,\right] \\
&\le \Pr\left[\, G_6^F \,\right] + \Pr\left[\, \mathsf{BD}_6 \,\right]
\end{aligned}$$

$$\begin{aligned}
\Pr\left[\, \mathsf{BD}_3 \,\right] &\le \sum_{i=1}^{i-1} \left(\frac{i-1}{p} + \frac{q_H + q_o}{p}\right) \\
&\le \frac{q_s^2 + 2q_s(q_H + q_o)}{2p}
\end{aligned}$$

$$\Pr\left[\, \mathsf{BD}_5 \,\right] \le \frac{q_s q_H}{p} \ .$$

So

$$\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) \le \Pr\left[\, G_6^F \,\right] + \frac{q_s^2 + 4q_s q_H + 2q_s q_o}{2p} \ .$$

Let A be the algorithm that on input $X \in G$, $h_1, \ldots, h_{q_H} \in \{0,1\}^k$ and coins $\rho = \rho_F||\sigma_1|| \ldots ||\sigma_{q_s}||\kappa_1|| \ldots ||\kappa_{q_s}$ where $\sigma_1, \ldots, \sigma_{q_s} \in \{0,1\}^k$ and $\kappa_1, \ldots, \kappa_{q_s} \in \mathbb{Z}_p$, runs F on input X and coins ρ_F. It lets $\sigma_1, \ldots, \sigma_{q_s}$ and $\kappa_1, \ldots, \kappa_{q_s}$ play the role of the quantities of the same name in $\mathbf{Initialize}$ of G_7. It answers F's queries to $\mathbf{ASign}$, $\mathbf{H}$, $\mathbf{Open}$ in the same way as G_7. When F outputs $(M, (\sigma, \kappa))$, algorithm A let

$$Y \leftarrow g^\kappa \cdot X^{-\sigma} \ ; \ \sigma' \leftarrow \mathbf{H}(X||Y||M) \ ; \ I \leftarrow \mathrm{Ind}(X||Y||M) \ .$$

where the call to $\mathbf{H}$ is answered as in G_7. If $M \in E$ or $\sigma \neq \sigma'$ then A returns $(0, q)$, else it returns $(I, (M, \sigma, \kappa, Y))$. Now consider the experiment where $\rho = \rho_F||\sigma_1|| \ldots ||\sigma_{q_s}||\kappa_1|| \ldots ||\kappa_{q_s}$ is chosen at random and then

$$x \leftarrow\!\!\$\ \mathbb{Z}_p \ ; \ h_1, \ldots, h_{q_H} \leftarrow\!\!\$\ \{0,1\}^k \ ; \ (I, s) \leftarrow\!\!\$\ A(g^x, h_1, \ldots, h_{q_H}; \rho).$$

Let acc be the probability that $I \neq 0$ in this experiment. Notice that if $M \notin E$ then $H[X||Y||M]$ was defined by an H-query $X||Y||M$ rather than by $\mathbf{Open}$, so $\mathrm{Ind}(X||Y||M) \in \{1, \ldots, q_H\}$. So

$$\mathrm{acc} = \Pr\left[\, G_7^F \,\right] \ .$$

Let *IG* be the algorithm that let $x \leftarrow\$ \mathbb{Z}_p$ and returns g^x. Let F_A be the algorithm of Lemma H.1 and let frk be defined as that.

How consider the experiment

$$x \leftarrow\$ \mathbb{Z}_p \; ; \; (b, s, s') \leftarrow F_A(g^x)$$

and assume $b = 1$. Let (I, s) and (I', s') be the output of A in the execution of F_A. Since $b = 1$ we have $I \neq 0$ and $I' \neq 0$, so we can parse $(M, Y, \sigma, \kappa) \leftarrow s$ and $(M', Y', \sigma', \kappa') \leftarrow s'$. The definition of A implies that Ind$(X||Y||M) = I$ and Ind$(X||Y'||M') = I'$. Now in the first execution of A it must be that $H[X||Y||M]$ was defined by an H-query of F rather than by $\mathbf{Open}$, and the response to the query was $\sigma = h_I$ which remains the value of $H[X||Y||M]$ thenceforth. Similarly in the second execution of A it must be that $H[X||Y'||M']$ was defined by an H-query of F rather than by $\mathbf{Open}$, and the response to the query was $\sigma' = h'_I$, which remains the value of $H[X||Y'||M']$ thenceforth. As a consequence $Y||M$ and $Y'||M'$ were determined by $x, h_1, \ldots, h_I(h_{I-1})$ (recall $I = I'$) and ρ and hence $Y||M = Y'||M'$. Now since $I \neq 0$ and $I' \neq 0$ we have

$$Y = g^{\kappa} \cdot X^{-\sigma} = g^{\kappa'} \cdot X^{-\sigma'} = Y'$$

and $\sigma \neq \sigma'$, so

$$x = g^{(\kappa - \kappa')a}$$

where $a = (\sigma - \sigma')^{-1} \mod p$. So F_A can easily be extended to an adversary B that on input X computes $\mathrm{DLog}(X)$ with probability frk. But by Lemma H.1 and the above

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F) &\leq \frac{q_s^2 + 4q_s q_H + 2q_s q_o}{2p} + \mathrm{acc} \\
&\leq \frac{q_s^2 + 4q_s q_H + 2q_s q_o}{2p} + \frac{q_H}{p} + \sqrt{q_H \cdot \mathrm{frk}}
\end{aligned}$$

The theorem follows.

I Proof of Theorem 7.2

Proof: We use games G_0, G_1, G_2 of Figure 14. We have

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) = 2 \cdot \Pr\left[\, G_0^A \,\right] - 1 \; . \tag{26}$$

Since games G_0 and G_1 are equivalent, we have

$$\Pr\left[\, G_0^A \,\right] = \Pr\left[\, G_1^A \,\right] \; . \tag{27}$$

Games G_1 and G_2 are identical until *bad.* Then based on Lemma 2.1, we have

$$\begin{aligned}
\Pr\left[\, G_1^A \,\right] &= \Pr\left[\, G_1^A \,\right] - \Pr\left[\, G_2^A \,\right] + \Pr\left[\, G_2^A \,\right] \\
&\leq \Pr\left[\, \mathsf{BD}_2 \,\right] + \Pr\left[\, G_2^A \,\right]
\end{aligned} \tag{28}$$

Combining (26), (27) and (28), we get

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) \leq 2 \cdot (\Pr\left[\, G_2^A \,\right] + \Pr\left[\, \mathsf{BD}_2 \,\right]) - 1 \; . \tag{29}$$

Note that in G_2, the challenge anonymous signature $H[X_b||Y||M]$ is set to be a random string with length k, so we have $\Pr\left[\,G_2^A\,\right] = \frac{1}{2}$ and thus

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(A) \le 2 \cdot \Pr\left[\,\mathsf{BD}_2\,\right]. \tag{30}$$

In addition, *bad* is set `true` when $H[X_b||Y||M]$ is already defined. Since Y is chosen randomly from group G of size p, we have

$$\Pr\left[\,\mathsf{BD}_2\,\right] \le \frac{q_H}{p}\ . \tag{31}$$

Equation (6) follows from (30) and (31).

J Proof of Theorem 7.3

Proof: Let $(X_0, X_1, M_0, M_1, \sigma, \kappa_0, \kappa_1)$ denote the output of A. If adversary A wins the game $\mathrm{UNAMB}_{\mathcal{AS}}$, then it must be that $X_0, X_1 \in G$ and $|\sigma| = k$ and $\kappa_0, \kappa_1 \in \mathbb{Z}_p$ and $\mathbf{H}(pk_0||Y_0||M_0) = \mathbf{H}(pk_1||Y_1||M_1) = \sigma$ where $Y_0 = g^{\kappa_0} X_0^{-\sigma}$ and $Y_1 = g^{\kappa_1} X_1^{-\sigma}$. But the probability that A can find a collision in RO H in q_H queries is at most $q_H^2/2^{k+1}$.

K Proof of Theorem 8.2

Proof: Consider games H_0, H_1 in Figure 15. We have

$$\mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(A) = 2\Pr\left[\,H_0^A\,\right] - 1\ .$$

H_1 and H_0 are identical until *bad*. By Lemma 2.1, we have

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(A) &= 2\Pr\left[\,H_0^A\,\right] - 1 \\
&= 2\Pr\left[\,H_1^A\,\right] + 2\Pr\left[\,H_0^A\,\right] - 2\Pr\left[\,H_1^A\,\right] - 1 \\
&= (2\Pr\left[\,H_1^A\,\right] - 1) + 2\Pr\left[\,\mathsf{BD}_1\,\right]
\end{aligned}$$

Lemma 8.1 gives us F such that

$$\Pr\left[\,\mathsf{BD}_1\,\right] \le \mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(F)\ .$$

It remains to design B so that

$$2(\Pr\left[\,H_1^A\,\right] - 1) \le n \cdot \mathbf{Adv}^{\mathrm{auf}}_{\mathcal{AS}}(B)\ . \tag{32}$$

Towards this end consider games $G_j, L_j (0 \le j \le n)$ of Figure 15. It is easy to see

$$2\Pr\left[\,H_1^A\,\right] - 1 = \Pr\left[\,L_n^A\,\right] - \Pr\left[\,L_0^A\,\right]\ . \tag{33}$$

The boxed code included in G_j is the key-swap that swaps the roles of $(pk_0, sk_0), (pk_1, sk_1)$ under certain conditions. However since $(pk_0, sk_0), (pk_1, sk_1)$ are independently chosen and only seen by A through the response to the **LR** query, swapping them has no effect visible to A, meaning

$$\Pr\left[\,G_j^A\,\right] = \Pr\left[\,L_j^A\,\right] (1 \le j \le n)\ . \tag{34}$$

We will design B so that

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{AS}}(B) = \frac{1}{n}(\Pr\left[\,G_n^A\,\right] - \Pr\left[\,G_0^A\,\right])\ . \tag{35}$$

Putiing together (33), (34) and (35) yields (32)and completes the proof.

Adversary B gets input $(pk_0, sk_0), (pk_1, sk_1)$. It picks $g \leftarrow\$ \{1, \ldots, n\}$ and then starts running A, responding to A's $\mathbf{LR}$ query via the following procedure

```
$\mathbf{LR}(M_0, M_1)$
If $(M_0[g] = 1 \wedge M_1[g] = 0)$ then
    $(pk, sk) \leftarrow (pk_0, sk_0)$ ; $(pk_0, sk_0) \leftarrow (pk_1, sk_1)$
    $(pk_1, sk_1) \leftarrow (pk, sk)$
For $i = 1, \ldots, g-1$ do $(\sigma_i, \kappa_i) \leftarrow\$ \mathsf{ASIG}(sk_{M_1[i]}, i)$
If $(M_0[g] = M_1[g])$ then $(\sigma_g, \kappa_g) \leftarrow\$ \mathsf{ASIG}(sk_{M_1[g]}, g)$
Else $(\sigma_g, \kappa_g) \leftarrow\$ \mathbf{CH}(g)$
For $i = g+1, \ldots, n$ do $(\sigma_i, \kappa_i) \leftarrow\$ \mathsf{ASIG}(sk_{M_0[i]}, i)$
$\sigma \leftarrow (0, \sigma_1 || \ldots || \sigma_n || pk_0 || pk_1)$
Return $\sigma$
```

Letting d denote the output of A adversary B returns d. Then letting b denote the challenge bit of $\mathrm{ANON}_{\mathcal{AS}}$. We claim that

$$\Pr\left[\, d = 1 \mid g = j \wedge b = 1 \,\right] = \Pr\left[\, G_j^A \,\right](1 \leq j \leq n)\,. \tag{36}$$

To justify this consider two cases. First, if $M_0[j] = M_1[j]$ then the code in B's simulated $\mathbf{LR}$ oracle is the same as in G_j. Second, if $M_0[j] \neq M_1[j]$, let $c = M_0[g]$. Then (σ_j, κ_j) is produced by $\mathbf{CH}(j)$ under $pk_{1\oplus c}$. (we use here that the key swap occurs if $c = 1$.) But $pk_{1\oplus c} = pk_{M_1[j]}$, since $c = M_0[j] = 1 \oplus M_1[j]$, so again this corresponds to G_j. On the other hand,

$$\Pr\left[\, d = 1 \mid g = j \wedge b = 0 \,\right] = \Pr\left[\, G_{j-1}^A \,\right](1 \leq j \leq n)\,. \tag{37}$$

To justify this consider two cases. First, if $M_0[j] = M_1[j]$ then the code in B's simulated $\mathbf{LR}$ oracle is equivalent to the one in G_{j-1} in this same case. Second, if $M_0[j] \neq M_1[j]$, let $c = M_0[j]$. Then (σ_j, κ_j) is produced by $\mathbf{CH}(j)$ under pk_c. (we use here that the key swap occurs if $c = 1$.) But $pk_c = pk_{M_0[j]}$, since $c = M_0[j]$, so this corresponds to G_{j-1}. Now from (36) and (37) we have

$$\begin{aligned}\mathbf{Adv}_{\mathcal{AS}}^{\mathrm{anon}}(B) &= \sum_{j=1}^{n} \frac{\Pr\left[\, G_j^A \,\right]}{n} - \frac{\Pr\left[\, G_{j-1}^A \,\right]}{n} \\ &= \frac{1}{n}(\Pr\left[\, G_n^A \,\right] - \Pr\left[\, G_0^A \,\right])\end{aligned}$$

which yields (35) as desired.

L Proof of Theorem 8.3

Proof: B runs A to obtain its output $(\sigma, (M_0, \omega_0), (M_1, \omega_1))$. Assume $\mathsf{CVF}(\sigma, (M_0, \omega_0)) = \mathsf{CVF}(\sigma, (M_1, \omega_1))$ $= 1$. B sets $(b, \sigma') \leftarrow \sigma$. If $b = 1$ then by definition of CVF it must be that $\sigma' = M_0 = \omega_0 = M_1 = \omega_1$, meaning $M_0 = M_1$, so A does not win and B returns $\perp$. If $b = 0$ then B parses σ' as $\sigma_1 || \ldots || \sigma_n || pk_0 || pk_1$ where $|\sigma_i| = l$, the latter being the length of a signature in $\mathcal{AS}$. Since keys also have a fixed length (as assumption we made in our signature syntax), the parsing process uniquely defines n from σ'. But then $\mathsf{CVF}(\sigma, (M_0, \omega_0)) = \mathsf{CVF}(\sigma, (M_1, \omega_1)) = 1$ implies that $n = |M_0| = |M_1|$ and $pk_0 \neq pk_1$. Now if A wins then it must be that $M_0 \neq M_1$, so let j be such that $M_0[j] \neq M_1[j]$. B further lets $\kappa_{c,1} || \ldots || \kappa_{c,n} \leftarrow \omega_c$ for $c = 0, 1$. B returns $(pk_0, pk_1, j, j, \sigma_j, \kappa_{0,j}, \kappa_{1,j})$.

```
**Initialize** // $G_0 - G_6$
$x \leftarrow\$ \mathbb{Z}_p$ ; $X \leftarrow g^x$ ; $E \leftarrow \emptyset$ ; $U \leftarrow \emptyset$ ; $i \leftarrow 0$
Return $X$

**ASign**$(M)$ // $\boxed{G_0}, G_1$
$i \leftarrow i+1$ ; $M_i \leftarrow M$
$\kappa_i \leftarrow\$ \mathbb{Z}_p$ ; $\sigma_i \leftarrow\$ \{0,1\}^k$ ; $Y_i \leftarrow g^{\kappa_i} X^{-\sigma_i}$
$S \leftarrow \{j : 1 \le j < i \wedge Y_j||M_j = Y_i||M_i\}$
If $S \neq \emptyset$ then $j \leftarrow\$ S$ ; $\sigma_i \leftarrow \sigma_j$ ; $\kappa_i \leftarrow \kappa_j$
Else if $H[X||Y_i||M_i]$ then
        $\sigma_i \leftarrow H[X||Y_i||M_i]$ ; $\kappa_i \leftarrow \mathrm{DLog}_g(Y_i) + x\sigma_i \mod p$
$\boxed{H[X||Y_i||M_i] \leftarrow \sigma_i}$
Return $\sigma_i$

**ASign**$(M)$ // $\boxed{G_2}, G_3$
$i \leftarrow i+1$ ; $M_i \leftarrow M$
$\kappa_i \leftarrow\$ \mathbb{Z}_p$ ; $\sigma_i \leftarrow\$ \{0,1\}^k$ ; $Y_i \leftarrow g^{\kappa_i} X^{-\sigma_i}$
$S \leftarrow \{j : 1 \le j < i \wedge Y_j||M_j = Y_i||M_i\}$
If $S \neq \emptyset$ then *bad* $\leftarrow$ true; $\boxed{j \leftarrow\$ S \text{ ; } \sigma_i \leftarrow \sigma_j \text{ ; } \kappa_i \leftarrow \kappa_j}$
Else if $H[X||Y_i||M_i]$ then *bad* $\leftarrow$ true
        $\boxed{\sigma_i \leftarrow H[X||Y_i||M_i] \text{ ; } \kappa_i \leftarrow \mathrm{DLog}_g(Y_i) + x\sigma_i \mod p}$
$\boxed{H[X||Y_i||M_i] \leftarrow \sigma_i}$
Return $\sigma_i$

**ASign**$(M)$ // $G_4, G_5$
$i \leftarrow i+1$ ; $M_i \leftarrow M$
$\kappa_i \leftarrow\$ \mathbb{Z}_p$ ; $\sigma_i \leftarrow\$ \{0,1\}^k$ ; $Y_i \leftarrow g^{\kappa_i} X^{-\sigma_i}$
Return $\sigma_i$

**ASign**$(M)$ // $G_6$
$i \leftarrow i+1$ ; $M_i \leftarrow M$ ; $\sigma_i \leftarrow\$ \{0,1\}^k$
Return $\sigma_i$

**ASign**$(M)$ // $G_7$
$i \leftarrow i+1$ ; $M_i \leftarrow M$
Return $\sigma_i$

**Finalize**$(M, (\sigma, \kappa))$ // $G_0 - G_6$
$Y \leftarrow g^{\kappa} X^{-\sigma}$ ; $\sigma' \leftarrow \mathbf{H}(X||Y||M)$
Return $(M \notin E \wedge H[X||Y||M] = \sigma)$

**Finalize**$(M, (\sigma, \kappa))$ // $G_7$
$Y \leftarrow g^{\kappa} X^{-\sigma}$ ; $\sigma' \leftarrow \mathbf{H}(X||Y||M)$ ; $I \leftarrow \mathrm{Ind}(X||Y||M)$
Return $(M \notin E \wedge \sigma = \sigma')$
```

```
**Initialize** // $G_7$
$x \leftarrow\$ \mathbb{Z}_p$ ; $X \leftarrow g^x$ ; $E \leftarrow \emptyset$ ; $c \leftarrow 0$ ; $i \leftarrow 0$
$h_1, \ldots, h_{q_H}, \sigma_1, \ldots, \sigma_{q_s} \leftarrow\$ \{0,1\}^k$
$\kappa_1, \ldots, \kappa_{q_s} \leftarrow\$ \mathbb{Z}_p$
Return $X$

**Open**$(j)$ // $G_0, \boxed{G_1}, \boxed{G_2}, \boxed{G_3}, \boxed{G_4}, \boxed{G_5}$
If $(j \le 0 \vee j > i)$ then return $\perp$
$E \leftarrow E \cup \{M_j\}$ ; $U \leftarrow U \cup \{j\}$
$\boxed{H[X||Y_j||M_j] \leftarrow \sigma_j}$
Return $\kappa_j$

**Open**$(j)$ // $G_6$
If $(j \le 0 \vee j > i)$ then return $\perp$
$\kappa_j \leftarrow\$ \mathbb{Z}_p$ ; $Y_j \leftarrow g^{\kappa_j} X^{-\sigma_j}$
$E \leftarrow E \cup \{M_j\}$
$H[X||Y_j||M_j] \leftarrow \sigma_j$
Return $\kappa_j$

**Open**$(j)$ // $G_7$
If $(j \le 0 \vee j > i)$ then return $\perp$
$Y_j \leftarrow g^{\kappa_j} X^{-\sigma_j}$ ; $E \leftarrow E \cup \{M_j\}$
$H[X||Y_j||M_j] \leftarrow \sigma_j$
Return $\kappa_j$

**H**$(x)$ // $G_1, G_2, G_3$
If $(H[x])$ then return $H[x]$
$X||Y||M \leftarrow x$ ; $H[x] \leftarrow\$ \{0,1\}^k$
$T \leftarrow \{j : 1 \le j \le i \wedge Y_j||M_j = Y||M \wedge j \notin U\}$
If $(T \neq \emptyset)$ then $j \leftarrow\$ T$ ; $H[x] \leftarrow \sigma_j$
Return $H[x]$

**H**$(x)$ // $\boxed{G_4}, G_5$
If $(H[x])$ then return $H[x]$
$X||Y||M \leftarrow x$ ; $H[x] \leftarrow\$ \{0,1\}^k$
$T \leftarrow \{j : 1 \le j \le i \wedge Y_j||M_j = Y||M \wedge j \notin U\}$
If $(T \neq \emptyset)$ then *bad* $\leftarrow$ true; $\boxed{j \leftarrow\$ T \text{ ; } H[x] \leftarrow \sigma_j}$
Return $H[x]$

**H**$(x)$ // $G_0, G_6$
If $(H[x])$ then return $H[x]$
$H[x] \leftarrow\$ \{0,1\}^k$
Return $H[x]$

**H**$(x)$ // $G_7$
If $(H[x])$ then return $H[x]$
$c \leftarrow c+1$ ; $H[x] \leftarrow h_c$ ; $\mathrm{Ind}(x) \leftarrow c$
Return $H[x]$
```

Figure 13: Game sequence used in proof of Theorem 6.2.

```
$\mathbf{Initialize}$ // $G_0$, $G_1$,$G_2$
$b \leftarrow_{\$} \{0,1\}$
$x_0 \leftarrow_{\$} \mathbb{Z}_p$ ; $x_1 \leftarrow_{\$} \mathbb{Z}_p$ ; $X_0 \leftarrow g^{x_0}$ ; $X_1 \leftarrow g^{x_1}$
Return $((x_0, X_0), (x_1, X_1))$

$\mathbf{CH}(M)$ // $\boxed{G_1}$,$G_2$
$\sigma \leftarrow_{\$} \{0,1\}^k$
$y \leftarrow_{\$} \mathbb{Z}_p$ ; $Y \leftarrow g^y$
If ($H[X_b||Y||M]$) then *bad* $\leftarrow$ true ;
    $\boxed{\sigma \leftarrow H[X_b||Y||M]}$
$H[X_b||Y||M] \leftarrow \sigma$
Return $\sigma$

$\mathbf{CH}(M)$ // $G_0$
$y \leftarrow_{\$} \mathbb{Z}_p$ ; $Y \leftarrow g^y$ ; $\sigma \leftarrow \mathbf{H}(X_b||Y||M)$
$\kappa \leftarrow y + \sigma x_b \mod p$
Return $\sigma$

$\mathbf{H}(x)$ // $G_0$, $G_1$,$G_2$
If ($H[x]$) Return $H[x]$
$H[x] \leftarrow_{\$} \{0,1\}^k$
Return $H[x]$

$\mathbf{Finalize}(d)$ // $G_0$, $G_1$, $G_2$
Return $(b = d)$
```

Figure 14: Game sequence used in proof of Theorem 7.2.

```
$\mathbf{Initialize}$ // $H_0$, $H_1$
$b \leftarrow_{\$} \{0,1\}$
$(pk_0, sk_0) \leftarrow_{\$}$ AKG()
$(pk_1, sk_1) \leftarrow_{\$}$ AKG()

$\mathbf{LR}(M_0, M_1)$ // $\boxed{H_0}$, $H_1$
For $i = 1$ to $n$
    $(\sigma_i, \kappa_i) \leftarrow \mathsf{ASIG}(sk_{M_b[i]}, i)$
$\sigma \leftarrow (0, \sigma_1|| \ldots ||\sigma_n||pk_0||pk_1)$
If $pk_0 = pk_1$ then *bad* $\leftarrow$ true; $\boxed{\sigma \leftarrow (1, M_b)}$
Return $\sigma$

$\mathbf{Finalize}(d)$ // $H_0$,$H_1$
Return $d = b$

$\mathbf{Initialize}$ // $G_j, L_j (0 \le j \le n)$
$(pk_0, sk_0) \leftarrow_{\$}$ AKG()
$(pk_1, sk_1) \leftarrow_{\$}$ AKG()

$\mathbf{LR}(M_0, M_1)$ // $\boxed{G_j}$,$L_j (0 \le j \le n)$
If $(M_0[j] = 1 \land M_1[j] = 0)$ then
    $\boxed{(pk, sk) \leftarrow (pk_0, sk_0)}$
    $\boxed{(pk_0, sk_0) \leftarrow (pk_1, sk_1)}$
    $\boxed{(pk_1, sk_1) \leftarrow (pk, sk)}$
For $i = 1, \ldots, j$ do $(\sigma_i, \kappa_i) \leftarrow_{\$} \mathsf{ASIG}(sk_{M_1[i]}, i)$
For $i = j+1, \ldots, n$ do $(\sigma_i, \kappa_i) \leftarrow_{\$} \mathsf{ASIG}(sk_{M_0[i]}, i)$
$\sigma \leftarrow (0, \sigma_1|| \ldots ||\sigma_n||pk_0||pk_1)$
Return $\sigma$

$\mathbf{Finalize}(d)$ // $G_j, L_j (0 \le j \le n)$
Return $d = 1$
```

Figure 15: Game sequence used in proof of Theorem 8.2.