Partial Signatures and their Applications

MIHIR BELLARE[*] SHANSHAN DUAN[†]

July 2009

Abstract

We introduce Partial Signatures, where a signer, given a message, can compute a "stub" which preserves her anonymity, yet later she, but nobody else, can complete the stub to a full and verifiable signature under her public key. We provide a formal definition requiring three properties, namely anonymity, unambiguity and unforgeability. We provide schemes meeting our definition both with and without random oracles. Our schemes are surprisingly cheap in both bandwidth and computation. We describe applications including anonymous bidding and betting.

Keywords: Signatures, anonymity, hash functions

[*] 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`. Supported in part by NSF grants CCF-0915675, CNS 1116800 and CNS 0904380.

[†] Work done while at the 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`. Supported in part by NSF grant CCF-0915675.

Contents

1 Introduction

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. Partial signatures allow her to do this as follows. Alice accompanies her bid bid_A with a "stub" σ. The stub cannot be verified given Alice's public verification key, and her anonymity is protected. When Alice's bid is pointed to as the winning one, she *and only she* can provide a de-anonymizer κ such that (σ, κ) is a full and normal signature which can be verified under the (certified) public verification key that she also now provides.

Partial signatures, unlike group [11, 4] and ring [19, 9] signatures, offer Alice a practical way to get anonymity in settings where she has no a priori knowledge of the "crowd" of people within which she wants to stay anonymous; they are suitable for anonymity in which the crowd, such as the set of all bidders in an auction, is dynamic and unknown to any individual bidder. Below we expand on the primitive and its formalization, and discuss solutions, both theoretical and practical. In Section 2 we explain in more depth how partial signatures differ from group [11, 4], ring [19, 9] and anonymous signatures [22, 14].

PARTIAL SIGNATURES. Signer Alice in a Partial Signature (PS) scheme generates for herself a secret signing key sk and a matching public verification key vk, and gets the latter certified in the usual way. Her verification key and certificate are then available to any potential verifier. So far, there is nothing different from a regular signature scheme.

The difference is that a signature of a message M under signing key sk is a pair (σ, κ). The stub σ is a "partial" signature, not by itself verifiable, provided in a first phase as a placeholder. Later the signer provides the de-anonymizer κ together with vk, and it is now possible to verify that (σ, κ) is a valid signature of M under vk. We now discuss the three security requirements, namely anonymity, unambiguity and unforgeability.

ANONYMITY. Briefly, the stub-creator's identity cannot be determined from the stub and message. To elaborate, with identities bound to verification keys, we imagine that the adversary has some a priori information about the potential verification key of the signer, for example that it belongs to some set S of keys. In the worst case, this set contains just two verification keys. Knowing both these keys, given a stub created under one of them, and given the message, the adversary has negligible advantage over guessing in determining under which of the two keys the stub was created.

An implication is that stubs, unlike full signatures, cannot be verified. Otherwise the adversary could test versus the two candidate verification keys to see which matched the stub.

UNAMBIGUITY. Anonymity by itself is easy to achieve: just let the stub be the empty string $\sigma = \varepsilon$ and let the de-anonymizer be the entire signature. But suppose Alice provides this trivial stub with her bid. Bob is watching, sees Alice's stub and bid, and sees that she wins. He now decides to claim the winning bid as his own by sending in his own public verification key vk' together with the de-anonymizer κ' consisting of Bob's signature of M under vk' which he can compute since he has the signing key sk' corresponding to vk'. Verification succeeds (recall in our example the de-anonymizer is the whole signature and the stub is empty) and Bob has now claimed the winning bid. Unambiguity prevents this by requiring that an adversary, given a stub σ under sk of a message M of her choice, is unable to create κ', vk' such that (σ, κ') verifies as a signature of m under vk'.

In the above attack, it sufficed for Bob to create vk' honestly, but in fact he is under no compulsion to do so. This is why we let the key vk' above be entirely under adversary control, leading to a strong security requirement. In fact our formal definition is stronger still, not even requiring that vk be honestly generated. The adversary simply provides verification keys vk_0, vk_1, messages M_0, M_1, de-anonymizers κ_0, κ_1 and a (single!) stub σ, and wins if (σ, κ_b) is a valid signature of M_b under vk_b for both $b = 0, 1$ *as long as the verification keys* vk_0, vk_1 *are distinct*. The adversary can generate vk_0, vk_1 any way it wants as long as they are different, and in particular might know both underlying secret keys.

Weaker requirements would in fact suffice for applications, but ours has the advantage of being very simply stated and turns out to be achievable without significant additional overhead, so we have adopted it. Unambiguity can be viewed as a signature analog of the robustness property of anonymous encryption defined in [2]. As there, it ensures that anonymity is not at the cost of authenticity.

UNFORGEABILITY. Strong as it is, our formulation of unambiguity does not imply standard unforgeability because the keys vk_0, vk_1 in the winning condition must be different. Unambiguity, thus, can be viewed as preventing "forgery" under an adversarially-modified verification key, something not part of the normal definition of a signature [15], and we will separately formulate an unforgeability requirement preventing forgery on the target verification key itself. However, here, too, something novel emerges, for we want to say that the adversary cannot forge a full signature on a message *even if it already knows a stub for this message*. That is, even if it has a piece of the signature, it should not be able to compute the rest. Furthermore we would like this to be true when the adversary has more capabilities than represented by the traditional signing oracle. We give it a partial signing oracle that returns stubs of messages of the adversary's choice. It can obtain corresponding de-anonymizers adaptively from an opening oracle and then wins if it forges on a stub for which it did not obtain the de-anonymizer. The ability to adaptively obtain de-anonymizers after seeing stubs is reminiscent of a selective-opening attack [12, 3] but we will be able in this context to provide solutions without heavy tools or additional overhead.

APPLICATIONS. Partial signatures are applicable in any setting where one wants anonymity in a first stage while reserving the capability, in a second stage, of identifying oneself and linking oneself to the first stage transaction. One such application is Alice placing a bid anonymously and then being able to claim it, if she so desires, at a later stage. Another is Alice betting anonymously and identifying herself only if she wins in order to claim the winnings. Author Alice could submit a paper to a conference anonymously, identifying herself only if the paper is accepted. Unambiguity prevents an adversary from, respectively, claiming the bid as its own; taking Alice's winnings; and claiming to be the author of Alice's paper.

ACHIEVING SECURITY. Having sketched the security requirements of a Partial Signature scheme we see that it asks for rather a lot more than a normal signature scheme. It requires a structured signature having two parts, one, the stub, not verifiable on its own but capable of being completed only by the stub creator. Unambiguity, a requirement on adversarially-chosen verification keys, is quite different from, and not in any way implied by, standard unforgeability, which pertains to honestly generated keys. This raises two questions. The first, a theoretical one, is whether security is achievable at all, and particularly without random oracles; the second a practical one, is whether one can find efficient solutions, now, if necessary, allowing random oracles. Let us address these questions in turn.

THEORETICAL CONSTRUCTION. We provide a simple, general transform of any standard signature scheme into a (secure, meaning meeting all three of our requirements) partial signature scheme. The transform uses as a tool any commitment scheme. This immediately yields constructions without random oracles, because standard signature schemes, as well as commitment schemes, without random oracles, are well known.

Proving unforgeability of our commitment-based partial signature scheme runs into the selective decommitment problem [12, 3]. 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 been able to prove this, and results in [12, 3] indicate that it is hard. In our particular setting, we are able to resolve the problem and prove security of our scheme by exploiting the fact that the privacy required for un-opened commitments is of a limited nature.

PRACTICAL CONSTRUCTIONS. The theoretical existence question thus settled, we turn to finding practical schemes. Here we should start by setting the stage. With regard to efficiency, we wish to minimize both computation and bandwidth. The motivation for the first is that public key cryptography is already considered expensive in many settings, and we do not wish to add a further computational burden.

Class	**Scheme**	**Sign**	**Ver**	$\lvert\sigma\rvert$	$\lvert\kappa\rvert$	**Assumption**
RH	RH-BLS	1 exp	1 pr	160	320	CDH
DH	DH-Sch	1 exp	2 exp	160	240	DL
DH	DH-GQ	1 exp	2 exp	160	2048	Factoring
SP	SP-Sch	1 exp	2 exp	80	160	DL

Figure 1: **Costs of our partial signature schemes.** For each scheme, we show the computational cost **Sign** of signing (this means generation of the full signature (σ, κ)); the computational cost **Ver** of verification; the bitlength $|\sigma|$ of a stub σ; the bitlength $|\kappa|$ 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 motivation for the second is that for wireless devices such as PDAs, cell phones, RFID chips and sensors, battery life is the main limitation. Here, communicating even one bit of data uses significantly more power than executing one 32-bit instruction. 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 stub and the de-anonymizer are as short as possible.

How well can we hope to do? Any partial signature scheme is, of course, a standard signature scheme. (The stub 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 a partial 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 partial 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 [21] based scheme, SP-ScH, has an 80 bit partial signature and a 160 bit de-anonymizer and has zero overhead, in *both* computation and bandwidth. Refer to Figure 1 for a summary of the characteristics of our schemes. We now discuss the schemes in more detail.

RH CONSTRUCTION. We can obtain fairly efficient schemes by instantiating the commitment scheme in our above-mentioned general transform by a random-oracle based randomized hash. The stub 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 [10] as the base signature scheme, and Figure 1 displays the characteristics of the resulting RH-BLS scheme.

DH CONSTRUCTION. We then consider 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 stub 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 [21], GQ [16] 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, 1], so there are numerous other choices as well, all quite efficient. (Note that the BLS scheme [10] 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.

Splitting construction. However, we can do even better. In identification-based signature schemes such as that of Schnorr [21], 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 partial signatures: we simply use σ as the stub, 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 [5]. 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.

Reverse connection. As indicated above, we have shown that one can build a partial 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 5 that any partial 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 [17, 18]. However, our transformation is direct and efficient.)

2 Related work

Anonymity means being lost in the crowd. Partial, group [11, 4] and ring [19, 9] signatures differ in how, when and by whom this crowd is defined. In group signatures the crowd is a pre-created group with corresponding management overhead and lack of flexibility; in ring signatures, the signer must explicitly pick the crowd at signing time and compute her signature as a function of it; but in a partial signature, the signer functions autonomously and obliviously of the crowd, which she does not need to know in order to compute a signature. Partial signatures are for settings where the crowd is ephemeral and dynamic and when the signer is potentially part of multiple crowds. Group and ring signatures are not applicable to anonymous bidding, where the crowd is ephemeral and not known in advance to any individual bidder. Anonymous signatures [22, 14] have the same intent as partial signatures but their definition does not lend itself well to the claimed applications and they lack security properties including unambiguity. Let us now look at these items in more detail.

Group signatures. In a group signature scheme [11, 4], all members of the group share a public verification key. A group manager provides each group member with a signing key, so that any group member can sign on behalf of the group. Anonymity means that, from the signature, one cannot tell which member of the group was the signer. In a partial signature, a signer generates her keys on her own and gets the verification key certified in the ordinary way. There is no explicit group. No manager or additional infrastructure is required.

There is no clear way to identify, in advance, the set of individuals who will bid in an open electronic auction, meaning form the crowd within which anonymity is sought. Even if one could, it is rather unlikely that this crowd is either able or willing to cooperate to form a group-signature group, which would involve finding a manager, getting a common public key, and setting up secure channels to the manager over which signing keys could be issued. Bidding crowds will be ephemeral, differing from bid to bid, making the overhead of group formation even more onerous. Partial signatures allow signers to sign in ignorance of the crowd and as members of ephemeral crowds not even defined at signing time.

Ring signatures. In a ring signature [19, 9], like in a partial signature, Alice generates her own keys and gets the verification key certified in the ordinary way. However, at signing time, she picks a set S of verification keys (her own key included in the set) that will form the crowd, and then computes her signature as a function of S. Anonymity means that, from the signature, one cannot tell which member of S was the signer. In a partial signature, in contrast the signing process does not have as input the crowd S. This reflects the needs of applications like anonymous bidding where the crowd will not be known in advance to an individual bidder.

Anonymous signatures. Introduced by Yang, Wong, Deng and Wang (YWDW) [22], anonymous signatures aim to address the same types of applications as partial signatures. We will argue however that they fall short in that the formulation does not lend itself well to applications and the security requirements are weak.

In an anonymous signature scheme, the recipient is provided with a full signature. The problem is that this is verifiable, and there are often only a few candidate verification keys, for example all bidders who eventually bid in the auction. By trial verification under the candidate verification keys, the signer can be determined. So how is one to get anonymity? The solution of YWDW [22] was, while giving the recipient the signature, to deny it the message. For this to prevent trial verification and provide anonymity, however, they require the message to be randomly chosen from a large space. Thus, they envisage providing the signature in a first phase and, in the second, providing the message and verification key.

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. For example, under the anonymous signatures approach, when Alice wishes to place a bid with value bid_A, she provides, at bidding time, her anonymous signature of bid_A, but not the message bid_A itself. However, the auctioneer needs to know the bid in order to determine the winner.

This problem is to some extent recognized in [22, 14]. 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 [14], but things are getting more complicated than seems desirable. The same issues arise with other applications mentioned in [22]. In summary, the whole "anonymity by message withholding" approach of YWDW just does not seem to map well to applications.

Partial signatures, in contrast, are a "anonymity by partial signature withholding" approach where the recipient is provided the message in full and no assumptions are made on its distribution. They better fit the applications for which anonymous signatures were intended.

The second weakness of the YWDW definition of anonymous signatures is that it fails to require 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. Partial signatures, in contrast, explicitly demand unambiguity, and all our schemes provide it.

With regard to schemes, YDWD [22] had no non-random oracle model solutions. The gap was filled by Fischlin [14], who provided some elegant constructions of anonymous signature schemes meeting the YWDW-definition, without random oracles. His constructions are based on extractors and use sophisticated techniques. In our case (partial signatures) we are able to get reasonably efficient solutions without random oracles and very efficient solutions with random oracles in relatively natural ways, an indication of a more usable definition.

Zhang and Imai [23] consider the case of anonymous signatures where messages have lower entropy. In work independent of ours, Saraswat and Yun [20] mount critiques on anonymous signatures similar to ours and suggest, instead of introducing hidden randomness to the message, to introduce hidden randomness to the signature. Instead of withholding part of the message, they too withhold part of the signature.

SCHEMES. Our partial signature schemes are simple and efficient and have minimal, even zero, overhead compared to standard signature schemes, which is appealing from the deployment perspective and is not true for any known group or ring signature schemes.

3 Definitions

NOTATION AND CONVENTIONS. We denote by $a = a_1 \| \cdots \| a_n$ an encoding of strings $a_1, \ldots, a_n$ from which the constituents are easily recoverable via $a_1 \| \cdots \| a_n \leftarrow a$. 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 $X = (x_1, x_2, \ldots, x_n)$ 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 [8] 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 3.1 [8] 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

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

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 (vk, 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 vk, 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 $\mathsf{SVF}(vk, s, M) = 1$ with probability 1 in the experiment $(vk, sk) \leftarrow\!\!\$\, \mathsf{SKG}()$; $s \leftarrow\!\!\$\, \mathsf{SIG}(sk, M)$. The standard unforgeability notion [15] is captured by the game EUF-CMA of Figure 8 in Appendix A.

PARTIAL SIGNATURES. A partial signature scheme $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ 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 stub and the second as the de-anonymizer. We propose three security properties: anonymity, unambiguity and unforgeability. The formal definitions are underlain by the games AN, UNAMB and UF shown in Figure 2. The corresponding adversary advantages are defined

Initialize $(vk, sk) \leftarrow\$ \mathsf{PKG}()$ $i \leftarrow 0\,;\ E \leftarrow \emptyset$ Return vk **Open**(j) If $(j \leq 0 \vee j > i)$ Return $\perp$ $E \leftarrow E \cup \{M_j\}$ Return κ_j **PSign**(M) $i \leftarrow i+1\,;\ M_i \leftarrow M$ $(\sigma_i, \kappa_i) \leftarrow\$ \mathsf{PSIG}(sk, M_i)$ Return σ_i **Finalize**$(M, (\sigma, \kappa))$ Return $(M \notin E \wedge \mathsf{PVF}(vk, M, (\sigma, \kappa)) = 1)$	**Initialize** $b \leftarrow\$ \{0,1\}$ $(vk_0, sk_0) \leftarrow\$ \mathsf{PKG}()$ $(vk_1, sk_1) \leftarrow\$ \mathsf{PKG}()$ Return $((vk_0, sk_0), (vk_1, sk_1))$ **CH**(M) $(\sigma, \kappa) \leftarrow\$ \mathsf{PSIG}(sk_b, M)$ Return σ **Finalize**(d) Return $(b = d)$	**Initialize** **Finalize**$(vk_0, vk_1, M_0, M_1, \sigma, \kappa_0, \kappa_1)$ $d_0 \leftarrow \mathsf{PVF}(vk_0, M_0, (\sigma, \kappa_0))$ $d_1 \leftarrow \mathsf{PVF}(vk_1, M_1, (\sigma, \kappa_1))$ Return $(d_0 = 1 \wedge d_1 = 1 \wedge vk_1 \neq vk_0)$

Figure 2: Games UF, AN and UNAMB used to define, respectively, unforgeability, anonymity and unambiguity of partial signature scheme $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$.

by $\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(A) = \Pr\left[\,\mathrm{UF}^{A}_{\mathcal{PS}}\,\right]$, $\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(A) = 2 \cdot \Pr\left[\,\mathrm{AN}^{A}_{\mathcal{PS}}\,\right] - 1$ and $\mathbf{Adv}^{\mathrm{unamb}}_{\mathcal{PS}}(A) = \Pr\left[\,\mathrm{UNAMB}^{A}_{\mathcal{PS}}\,\right]$ respectively.

In game UF, an adversary F can query the oracle **PSign** to get a stub 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 **PSign** or M was queried to **PSign** but the signature returned was not opened.

The formalization of anonymity follows [4]. The adversary not only gets target public keys vk_0 and vk_1 but also knows the corresponding secret keys sk_0 and sk_1. Via the **CH** oracle, it can obtain a stub, 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 a stub σ of some message M_0 under her public key vk_0. Unambiguity ensures that only Alice can open σ, by requiring that an adversary be unable to produce a public key vk_1, message M_1 and de-anonymizer κ_1 such that $\mathsf{PVF}(vk_1, M_1, (\sigma, \kappa_1)) = 1$ but $vk_0 \neq vk_1$. Actually the requirement is stronger, preventing even Alice herself from a 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 a stub, "change" the message or identity from which it "originated".

4 Constructions

The StC Construction. We describe a general transform of any signature scheme into a partial one based on the following simple idea: the stub 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 partial 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 that a commitment scheme $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ consists of two algorithms.

Alg $\mathsf{PKG}()$
$(vk, sk) \leftarrow_{\$} \mathsf{SKG}()$
Return (vk, sk)
	Alg $\mathsf{PSIG}(sk, M)$
	$s \leftarrow_{\$} \mathsf{SIG}(sk, M)$
	$(\sigma, \omega) \leftarrow_{\$} \mathsf{CMT}(s\|\|vk)$
	$\kappa \leftarrow (s, \omega)$
	Return σ
		Alg $\mathsf{PVF}(vk, M, (\sigma, \kappa))$
		$(s, \omega) \leftarrow \kappa$
		If $(\mathsf{CVF}(\sigma, s\|\|vk, \omega) = 1)$ then
		  If $(\mathsf{SVF}(vk, s, M) = 1)$
		    then Return 1
		Return 0

Figure 3: Algorithms defining partial signature scheme $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ derived via the StC transform from base signature scheme $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ and commitment scheme $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$.

Alg $\mathsf{PKG}()$ $(vk, sk) \leftarrow_{\$} \mathsf{SKG}()$ Return (vk, sk)	Alg $\mathsf{PSIG}^H(sk, M)$ $s \leftarrow_{\$} \mathsf{SIG}(sk, M)$; $\omega \leftarrow_{\$} \{0,1\}^k$ $\sigma \leftarrow_{\$} H(\omega\|\|s\|\|vk)$ $\kappa \leftarrow (\omega, s)$ Return (σ, κ)	Alg $\mathsf{PVF}^H(vk, M, (\sigma, \kappa))$ $(\omega, s) \leftarrow \kappa$ If $(H(\omega\|\|s\|\|vk) = \sigma \wedge \|\omega\| = k)$ then If $(\mathsf{SVF}(vk, s, M) = 1)$ then return 1 Return 0

Figure 4: Algorithms defining partial signature scheme $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ derived via the RH transform from base signature scheme $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$.

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 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 partial signature scheme $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ whose constituent algorithms are defined in Figure 3. The following theorem says that (1) if $\mathcal{DS}$ is unforgeable and $\mathcal{CMT}$ is hiding then $\mathcal{PS}$ is unforgeable (2) If $\mathcal{CMT}$ is hiding then $\mathcal{PS}$ is anonymous, and (3) if $\mathcal{CMT}$ is binding then $\mathcal{PS}$ is unambiguous. The proof is in Appendix B.

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 $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ be the partial signature scheme constructed from $\mathcal{DS}$ and $\mathcal{CMT}$ as in Figure 3. Then we have:

1. UNFORGEABILITY: Let F be an adversary against the unforgeability of $\mathcal{PS}$ making $q \geq 1$ queries to oracle **PSign**. Then there exist adversaries A, B such that $\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F) \leq 2q \cdot \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A) + q \cdot \mathbf{Adv}^{\mathrm{hd}}_{\mathcal{CMT}}(B)$. Furthermore, the running times of A, B are the same as the running time of F, and A makes q queries to its **Sign** oracle.

2. ANONYMITY: Let A be an adversary against the unambiguity of $\mathcal{PS}$. Then there exists an adversary B such that $\mathbf{Adv}^{\mathrm{unamb}}_{\mathcal{PS}}(A) \leq \mathbf{Adv}^{\mathrm{bnd}}_{\mathcal{CMT}}(B)$. Furthermore, the running time of B is that of A.

3. UNAMBIGUITY: Let A be an adversary against the anonymity of $\mathcal{PS}$ that makes one query to oracle **CH**. Then there exists adversary B such that $\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(A) \leq \mathbf{Adv}^{\mathrm{hd}}_{\mathcal{CMT}}(B)$. Furthermore, the running time of B is that of A. ▌

THE RH CONSTRUCTION. The Randomized Hash (RH) construction is the result of instantiating the commitment scheme of the StC construction with the RO-model commitment scheme $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ defined as follows. Algorithm $\mathsf{CMT}^H(M)$ picks $\omega \leftarrow_{\$} \{0,1\}^k$ and returns $H(\omega\|\|M)$ as the commitment, where H is the RO. Algorithm $\mathsf{CVF}^H(\sigma, M, \omega)$ lets $\sigma' \leftarrow H(\omega\|\|M)$. If $|\omega| \neq k$ then it returns 0. Else if $\sigma = \sigma'$ then it returns 1 else it returns 0. Figure 4 depicts the algorithms of partial signature scheme $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ obtained from the StC 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 $\mathcal{PS}$ 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 stub 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 then the RO H of Figure 4 must be different and independent. This can be ensured by domain seperation as discussed in [6]. This issue arises also below and should be addressed in the same way.

The DH Construction. Base signature schemes such as Schnorr [21], GQ [16] 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 a partial one $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ using a RO $H\colon \{0,1\}^* \to \{0,1\}^k$, as shown in Figure 5. For the analysis, we make the following definition. Let $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ be a digital signature scheme. The min-entropy $\mathsf{H}_\infty(\mathcal{DS})$ of $\mathcal{DS}$ is defined by the equation

$$2^{-\mathsf{H}_\infty(\mathcal{DS})} = \max_{(vk,sk),\overline{s},M} \Pr\left[\, \overline{s} = s : s \leftarrow\$\, \mathsf{SIG}(M, sk) \,\right]$$

where the maximum is over all (vk, sk) that might be output by SKG, all strings $\overline{s}$, and all messages M. For example, the Schnorr (Sch) scheme [21] over a group of order p has min-entropy $\lg(p)$. A deterministic scheme such as FDH [7] or BLS [10], however, has min-entropy 0. The DH-Sch scheme has bandwidth overhead 160 bits as compared to 320 bits for RH-Sch.

The following theorem says that the partial signature scheme of Figure 5 is secure in the RO model assuming a secure, high entropy base signature scheme. The proof is in Appendix C.

Theorem 4.2 Let $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$ be a digital signature scheme. Let $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ be the partial signature scheme constructed as in Figure 5. Let k be the output length of the RO H in the scheme. Then we have:

1. Unforgeability: Let F be an adversary against the unforgeability of $\mathcal{PS}$, making q_s queries to oracle **PSign**, q_H queries to random oracle **H** and q_o queries to oracle **Open**. Then there exists an adversary A such that $\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F) \le \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A) + q_s(q_s + 4(q_H + q_o)) \cdot 2^{-1-\mathsf{H}_\infty(\mathcal{DS})}$. Furthermore, the running time of A is that of F and A makes q_o queries to its **Sign** oracle.

2. Anonymity: Let A be an adversary against the anonymity of $\mathcal{PS}$ making q_H queries to random oracle **H** and one query to oracle **CH**. Then $\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(A) \le 2q_H \cdot 2^{-\mathsf{H}_\infty(\mathcal{DS})}$.

3. Unambiguity: Let A be an adversary against the unambiguity of $\mathcal{PS}$ making q_H queries to random oracle **H**. Then we have $\mathbf{Adv}^{\mathrm{unamb}}_{\mathcal{PS}}(A) \le q_H^2 \cdot 2^{-k-1}$. ∎

We remark that the proof shows that for unambiguity it suffices for the hash function to be collision resistant rather than a random oracle.

The Splitting Construction. The splitting construction of a partial signature is based on the Schnorr protocol [21], recalled in Figure 6, and a hash function. 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 stub while the message in the third move is viewed as a de-anonymizer.

Alg $\mathsf{PKG}()$	Alg $\mathsf{PSIG}^H(sk, M)$	Alg $\mathsf{PVF}^H(vk, M, (\sigma, \kappa))$
$(vk, sk) \leftarrow_{\$} \mathsf{SKG}()$	$s \leftarrow_{\$} \mathsf{SIG}(sk, M)$; $\sigma \leftarrow_{\$} H(s \| vk)$	$s \leftarrow \kappa$
Return (vk, sk)	$\kappa \leftarrow s$	If $(H(s \| vk) = \sigma) \wedge (\mathsf{SVF}(vk, s, M) = 1)$ then
	Return (σ, κ)	Return 1
		Return 0

Figure 5: Algorithms defining partial signature scheme $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ derived via the DH transform applied to high-entropy base signature scheme $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$.

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

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

Figure 6: At the top is the Schnorr identification protocol. Below are the algorithms defining partial signature scheme $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ derived from this protocol via the splitting construction. Here G is a group of prime order p and g is a generator of G.

The associated partial signature scheme $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ 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 partial 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 a partial signature scheme with an 80-bit stub and a 160 bit de-anonymizer for a 240-bit full signature. Our proof will exploit the general forking lemma of [5], recalled in Appendix D.

To discuss security we first recall the Discrete Logarithm Assumption. 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

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

denote the advantage of an adversary A in attacking the discrete logarithm (dl) problem. The proof of the following theorem is in Appendix D.

Theorem 4.3 Let G be a group of prime order p and let g be a generator of G. Let $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ be the splitting-based partial signature scheme constructed in Figure 6. Let the range of the RO H in the scheme be $\{0,1\}^k \subseteq \mathbb{Z}_p$. Then we have:

Alg $\mathsf{CMT}(M)$	Alg $\mathsf{CVF}(\sigma, M, \omega)$
$(vk_0, sk_0) \leftarrow\$ \mathsf{PKG}()$	$(b, \sigma') \leftarrow \sigma$
$(vk_1, sk_1) \leftarrow\$ \mathsf{PKG}()$	If $(b = 1)$ then
If $(vk_0 = vk_1)$ then $bad \leftarrow \mathtt{true}$	If $(\sigma' = M \wedge \omega = M)$ then return 1
$n \leftarrow \|M\|$	Else return 0
For $i = 1$ to n	Else
$(\sigma_i, \kappa_i) \leftarrow\$ \mathsf{PSIG}(sk_{M[i]}, i)$	$\kappa_1 \|\| \ldots \|\| \kappa_n \leftarrow \omega$
$\sigma \leftarrow (0, \sigma_1 \|\| \ldots \|\| \sigma_n \|\| vk_0 \|\| vk_1)$	$\sigma_1 \|\| \ldots \|\| \sigma_n \|\| vk_0 \|\| vk_1 \leftarrow \sigma'$
$\omega \leftarrow \kappa_1 \|\| \ldots \|\| \kappa_n$	If $(vk_0 = vk_1)$ then return 0
If $bad = \mathtt{true}$ then $\sigma \leftarrow (1, M)$; $\omega \leftarrow M$	For $i = 1$ to n $d_i \leftarrow \mathsf{PVF}(vk_{M[i]}, i, (\sigma_i, \kappa_i))$
Return (σ, ω)	Return $d_1 \wedge \ldots \wedge d_n$

Figure 7: CMT construction from PS.

1. <u>UNFORGEABILITY:</u> Let F be an adversary against the unforgeability of $\mathcal{PS}$, making q_s queries to oracle **PSign**, q_H queries to random oracle **H** and having running time t_F. Then there exists an algorithm B against the discrete logarithm problem such that

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F) \leq \frac{q_s^2 + 4q_s q_H + 2q_s q_o}{2p} + \frac{q_H}{p} + \sqrt{q_H \cdot \mathbf{Adv}^{\mathrm{dl}}_{G,g}(B)} \; .$$

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

2. <u>ANONYMITY:</u> Let A be an adversary against the anonymity of $\mathcal{PS}$ making q_H queries to random oracle **H** and one query to oracle **LR**. Then $\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(A) \leq 2q_H/p$.

3. <u>UNAMBIGUITY:</u> Let A be an adversary against the unambiguity of $\mathcal{PS}$ making q_H queries to random oracle **H**. Then $\mathbf{Adv}^{\mathrm{unamb}}_{\mathcal{PS}}(A) \leq q_H^2/2^{k+1}$. ▮

5 A Reverse Connection

From the primitive definitions, we can see that partial signatures (PS) and commitment schemes (CMT) share something in common. Firstly, PS hide the identity of the signer while CMT hide the committed message. Secondly, in the PS 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 partial 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, especially an efficient one. Our transformation, which provides a direct and efficient conversion from PS to CMT, is depicted in Figure 7.

SECURITY OF OUR CONSTRUCTION. We prove that if the given partial 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 5.1 to bound the probability that such public key collision happens.

procedure Initialize // $\mathrm{PKColl}_{\mathcal{PS}}$
$(vk_0, sk_0) \leftarrow\!\!\$\ \mathsf{PKG}()$
$(vk_1, sk_1) \leftarrow\!\!\$\ \mathsf{PKG}()$
Return $(vk_0 = vk_1)$

Lemma 5.1 Let $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ be a partial signature scheme. Then there is an adversary F against the unforgeability of $\mathcal{PS}$ such that $\Pr[\mathrm{PKColl}_{\mathcal{PS}}] \leq \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F)$. The running time of F is that of PKG and F makes no oracle queries.

Proof: On input vk F let $vk_0 \leftarrow vk$ and $(vk_1, sk_1) \leftarrow\!\!\$\ \mathsf{PKG}$. It let M be any message, for example $M = 0$. It lets $(\sigma, \kappa) \leftarrow\!\!\$\ \mathsf{PSIG}(sk, M)$ and returns $(M, (\sigma, \kappa))$. If $vk_1 = vk_0$, then it wins the game $\mathrm{UF}_{\mathcal{PS}}$, so we have $\Pr[\mathrm{PKColl}_{\mathcal{PS}}] \leq \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F)$.

Theorem 5.2 Let $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ be a partial signature scheme and $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ the commitment scheme constructed from $\mathcal{PS}$ as in Figure 7. Let A be an adversary against the hiding property of $\mathcal{CMT}$, making one query to oracle **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 **CH** and adversary F making no queries such that

$$\mathbf{Adv}^{\mathrm{hd}}_{\mathcal{CMT}}(A) \leq n \cdot \mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(B) + 2 \cdot \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F) .$$

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

The proof is in Appendix E.

Theorem 5.3 Let $\mathcal{PS} = (\mathsf{PKG}, \mathsf{PSIG}, \mathsf{PVF})$ be a partial signature scheme and $\mathcal{CMT} = (\mathsf{CMT}, \mathsf{CVF})$ the commitment scheme constructed from $\mathcal{PS}$ as in Figure 7. Let A be an adversary against the binding property of $\mathcal{CMT}$. Then there exists an adversary B such that

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

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

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

References

[1] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, Apr. / May 2002. 5

[2] M. Abdalla, M. Bellare, and G. Neven. Robust encryption. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, Feb. 2010. 4

[3] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Apr. 2009. 4

[4] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, May 2003. 3, 6, 9

[5] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 390–399. ACM Press, Oct. / Nov. 2006. 6, 12, 20

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 11

[7] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, May 1996. 11

[8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. 8

[9] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 60–79. Springer, Mar. 2006. 3, 6

[10] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, Sept. 2004. 5, 11

[11] D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Apr. 1991. 3, 6

[12] C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534. IEEE Computer Society Press, Oct. 1999. 4

[13] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Aug. 1987. 5, 11

[14] M. Fischlin. Anonymous signatures made easy. In T. Okamoto and X. Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 31–42. Springer, Apr. 2007. 3, 6, 7

[15] 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, Apr. 1988. 4, 8

[16] L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 123–128. Springer, May 1988. 5, 11

[17] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 6

[18] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. 6

[19] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Dec. 2001. 3, 6

[20] V. Saraswat and A. Yun. Anonymous signatures revisited. Cryptology ePrint Archive, Report 2009/307, 2009. `http://eprint.iacr.org/`. 8

[21] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. 5, 6, 11

[22] G. Yang, D. S. Wong, X. Deng, and H. Wang. Anonymous signature schemes. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 347–363. Springer, Apr. 2006. 3, 6, 7

[23] R. Zhang and H. Imai. Strong anonymous signatures. In M. Yung, P. Liu, and D. Lin, editors, *INSCRYPT08*, volume 5487 of *LNCS*, pages 60–71. Springer, Dec. 2008. 8

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.

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

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

$\mathbf{Finalize}(d)$
Return $(b = d)$

Initialize

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

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

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

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

Figure 8: Game HD in the left used to define hiding and game BND 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 $\mathcal{DS} = (\mathsf{SKG}, \mathsf{SIG}, \mathsf{SVF})$.

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

$\mathbf{Open}(j)$ // 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 κ_j

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

$\mathbf{PSign}(M)$ // 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 $s_i \leftarrow\!\!\$\ \mathsf{SIG}(sk, M_i)$
$(\sigma_i, \omega_i) \leftarrow\!\!\$\ \mathsf{CMT}(s_i \| vk)$
$\kappa_i \leftarrow (s_i, \omega_i)$
Return σ_i

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

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

$\mathbf{Finalize}(M, (\sigma, \kappa))$ // G_2, G_3, G_4
Return $(M = M_g \wedge M \notin E \wedge \mathsf{PVF}(vk, M, (\sigma, \kappa)) = 1)$

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 advantage of an adversary A in attacking the hiding property is

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

where game HD 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

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

where game BND is in Figure 8.

B Proof of Theorem 4.1

Proof of Part 1.: 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 **PSign** 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

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F) \;\le\; \Pr\left[\,G_0^F\,\right] + \Pr\left[\,G_1^F\,\right] . \tag{1}$$

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

$$\Pr\left[\,G_1^F\,\right] \le q \cdot \Pr\left[\,G_2^F\,\right] . \tag{2}$$

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 3.1, we have

$$\Pr\left[\,G_2^F\,\right] \;=\; \Pr\left[\,G_2^F \wedge \overline{\mathsf{BD}}_2\,\right] \;=\; \Pr\left[\,G_3^F \wedge \overline{\mathsf{BD}}_3\,\right] . \tag{3}$$

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

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS},F}(k) \;\le\; \Pr\left[\,G_0^F\,\right] \;+\; q \cdot \Pr\left[\,G_3^F \wedge \overline{\mathsf{BD}}_3\,\right] . \tag{4}$$

We will build A_0, A_1, B so that

$$\Pr\left[\,G_0^F\,\right] \;\le\; \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A_0) \tag{5}$$

$$\Pr\left[\,G_3^F \wedge \overline{\mathsf{BD}}_3\,\right] - \Pr\left[\,G_4^F \wedge \overline{\mathsf{BD}}_4\,\right] \;\le\; \mathbf{Adv}^{\mathrm{hd}}_{\mathcal{CMT}}(B) \tag{6}$$

$$\Pr\left[\,G_4^F \wedge \overline{\mathsf{BD}}_4\,\right] \;\le\; \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A_1) \tag{7}$$

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 vk pick $c \leftarrow_{\$} \{0,1\}$ and run $A_c(vk)$. Then

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A) \;=\; \frac{1}{2}\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A_0) + \frac{1}{2}\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A_1) . \tag{8}$$

Part **1.** of Theorem 4.1 follows from (4), (5) (6), (7) and (8). We proceed to describe A_0, A_1, B.

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

$$S \leftarrow \emptyset\,;\; E \leftarrow \emptyset\,;\; i \leftarrow 0\,;\; j \leftarrow 0\,;\; g \leftarrow_{\$} \{1, \ldots, q\} . \tag{9}$$

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

procedure PSign(M)
$i \leftarrow i+1\,;\; M_i \leftarrow M\,;\; S \leftarrow S \cup \{M_i\}\,;\; s_i \leftarrow_{\$} \mathbf{Sign}(M)$
$(\sigma_i, \omega_i) \leftarrow_{\$} \mathsf{CMT}(s_i || vk)\,;\; \kappa_i \leftarrow (s_i, \omega_i)$
Return σ_i

A_0 answers F's queris to **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 **Initialize** procedure of G_3, thereby defining for itself the parameters vk, sk, S, E, i, j, g. It then starts running F on vk. It answers F's queries to **PSign** using the following procedure:

procedure PSign(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 $s_0 \leftarrow_{\$} \{0,1\}^l\,;\; \sigma_i \leftarrow \mathbf{LR}(s_0||vk, s_i||vk)$

Else $(\sigma_i, \omega_i) \leftarrow\$ \mathsf{CMT}(s_i||vk)$; $\kappa_i \leftarrow (s_i, \omega_i)$
Return σ_i

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

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

Subtracting, we get

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

Adversary A_1 gets input vk and then does the initializations (9). It then runs $F(vk)$. It answers F's queries to **PSign** using the following procedure:

procedure **PSign**(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 $s_i \leftarrow\$ \mathbf{Sign}(M)$
$(\sigma_i, \omega_i) \leftarrow\$ \mathsf{CMT}(s_i||vk)$; $\kappa_i \leftarrow (s_i, \omega_i)$
Return σ_i

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

Proof of Part 2.: Adversary B begins with $(vk_i, sk_i) \leftarrow\$ \mathsf{PKG}()$ for $i = 0, 1$. It then runs $A((vk_0, sk_0), (vk_1, sk_1))$ and answers A's queries to **CH** using the following procedure:

procedure **CH**(M)
$s_0 \leftarrow\$ \mathsf{SIG}(sk_0, M)$; $s_1 \leftarrow\$ \mathsf{SIG}(sk_1, M)$; $\sigma \leftarrow \mathbf{LR}(s_0||vk_0, s_1||vk_1)$
Return σ

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

$$\Pr\left[\, \mathrm{HD}^B_{\mathcal{CMT}} \mid b = 1 \,\right] = \Pr\left[\, \mathrm{AN}^A_{\mathcal{PS}} \mid b = 1 \,\right] \text{ and } \Pr\left[\, \mathrm{HD}^B_{\mathcal{CMT}} \mid b = 0 \,\right] = \Pr\left[\, \mathrm{AN}^A_{\mathcal{PS}} \mid b = 0 \,\right]$$

from which Part **2.** of Theorem 4.1 follows. ▮

Proof of Part 3.: Adversary B runs A to get $(vk_0, vk_1, M_0, M_1, \sigma, \kappa_0, \kappa_1)$. It lets $(s_0, \omega_0) \leftarrow \kappa_0$ and $(s_1, \omega_1) \leftarrow \kappa_1$. Adversary B then outputs $\sigma, (s_0||vk_0, \omega_0), (s_1||vk_1, \omega_1)$. ▮

C Proof of Theorem 4.2

Proof of Part 1.: We refer to the games of Figure 10. Game G_0 is equivalent to $\mathrm{UF}_{\mathcal{PS}}$, so

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

Game G_1 omits the boxed code in **PSign**, meaning $H[s_i||vk]$ is not assigned σ_i at this point. Instead the assignment is delayed, being done by $H(x)$ or **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 3.1

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

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

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

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

PSign(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 σ_i

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

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

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

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

H(x) // G_1, G_2, G_3
If $(H[x])$ Return $H[x]$
$s||vk \leftarrow x$; $H[x] \leftarrow\$ \{0,1\}^k$
$T \leftarrow \{j : 1 \leq j \leq i \wedge s = s_j \wedge j \notin U\}$
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 $H[x]$
$s||vk \leftarrow x$; $H[x] \leftarrow\$ \{0,1\}^k$
$T \leftarrow \{j : 1 \leq j \leq i \wedge s = s_j \wedge j \notin U\}$
If $(T \neq \emptyset)$ then $bad \leftarrow \mathtt{true}$; $\boxed{j \leftarrow\$ T \text{ ; } H[x] \leftarrow \sigma_j}$
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 4.2.

G_3 and G_4 are equivalent and G_4 and G_5 are identical until bad, so by Lemma 3.1

$$\Pr\left[\,G_3^F\,\right] = \Pr\left[\,G_4^F\,\right] = \Pr\left[\,G_5^F\,\right] + \Pr\left[\,G_4^F\,\right] - \Pr\left[\,G_5^F\,\right] \leq \Pr\left[\,G_5^F\,\right] + \Pr\left[\,\mathsf{BD}_5\,\right].$$

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

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F) \leq \Pr\left[\,G_6^F\,\right] + \Pr\left[\,\mathsf{BD}_3\,\right] + \Pr\left[\,\mathsf{BD}_5\,\right]. \tag{10}$$

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

$$\Pr\left[\,G_6^F\,\right] \leq \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(A) \tag{11}$$

Now

$$\Pr\left[\,\mathsf{BD}_3\,\right] \;\leq\; \sum_{i=1}^{q_s}\left(\frac{i-1}{2^{\mathsf{H}_\infty(\mathcal{DS})}} + \frac{q_H + q_o}{2^{\mathsf{H}_\infty(\mathcal{DS})}}\right) \;=\; \frac{q_s(q_s-1) + 2q_s(q_H+q_o)}{2^{1+\mathsf{H}_\infty(\mathcal{DS})}}\,. \tag{12}$$

Initialize // G_0, G_1,G_2	**CH**(M) // $\boxed{G_1}$,G_2
$b \leftarrow\$ \{0,1\}$	$s \leftarrow\$ \mathsf{SIG}(sk_b, M)$; $\sigma \leftarrow\$ \{0,1\}^k$
$(vk_0, sk_0) \leftarrow\$ \mathsf{SKG}()$	If $(H[s\|\|vk_b])$ then $bad \leftarrow \texttt{true}$; $\boxed{\sigma \leftarrow H[s\|\|vk_b]}$
$(vk_1, sk_1) \leftarrow\$ \mathsf{SKG}()$	$H[s\|\|vk_b] \leftarrow \sigma$
Return $((vk_0, sk_0), (vk_1, sk_1))$	Return σ
CH(M) // G_0	**H**(x) // G_0, G_1,G_2
$s \leftarrow\$ \mathsf{SIG}(sk_b, M)$; $\sigma \leftarrow \mathbf{H}(s\|\|vk_b)$	If $(H[x])$ Return $H[x]$
Return σ	$H[x] \leftarrow\$ \{0,1\}^k$
Finalize(d) // G_0, G_1, G_2	Return $H[x]$
Return $(b = d)$	

Figure 11: Game sequence used in proof of Theorem 4.2.

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

$$\Pr[\,\mathsf{BD}_5\,] \leq \frac{q_s q_H}{2^{\mathsf{H}_\infty(\mathcal{DS})}} . \tag{13}$$

Putting together (10), (11), (12) and (13) completes the proof. ▮

Proof of Part 2.: We use games G_0, G_1, G_2 of Figure 11. So we have

$$\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(A) = 2 \cdot \Pr\left[\,G_0^A\,\right] - 1 . \tag{14}$$

Games G_0 and G_1 are equivalent, and G_1 and G_2 are identical until *bad* so by Lemma 3.1 we have

$$\Pr\left[\,G_0^A\,\right] = \Pr\left[\,G_1^A\,\right] = \Pr\left[\,G_1^A\,\right] - \Pr\left[\,G_2^A\,\right] + \Pr\left[\,G_2^A\,\right] \leq \Pr[\,\mathsf{BD}_2\,] + \Pr\left[\,G_2^A\,\right] . \tag{15}$$

Combining (14) and (15), we get

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

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

$$\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(A) \leq 2 \cdot \Pr[\,\mathsf{BD}_2\,] . \tag{16}$$

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

$$\Pr[\,\mathsf{BD}_2\,] \leq q_H \cdot 2^{-\mathsf{H}_\infty(\mathcal{DS})} . \tag{17}$$

Part **2.** of Theorem 4.2 follows from (16) and (17). ▮

Proof of Part 3.: Let $(vk_0, vk_1, M_0, M_1, \sigma, \kappa_0, \kappa_1)$ denote the output of A. Let $s_0 \leftarrow \kappa_0$ and $s_1 \leftarrow \kappa_1$. If A wins the game $\mathrm{UNAMB}_{\mathcal{PS}}$, then we have $H(s_0||vk_0) = H(s_1||vk_1) = \sigma$ but $vk_0 \neq vk_1$, meaning that we have a collision for H. Since A makes q_H queries to H we have Part **3.** of Theorem 4.2. ▮

D Proof of Theorem 4.3

Before giving the security proof, we first recall the general forking lemma [5], which will be used later.

Lemma D.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

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

PSign(M) // $\boxed{G_0}$, G_1
$i \leftarrow i+1$; $M_i \leftarrow M$
$\kappa_i \leftarrow_{\$} \mathbb{Z}_p$; $\sigma_i \leftarrow_{\$} \{0,1\}^k$; $Y_i \leftarrow g^{\kappa_i} X^{-\sigma_i}$
$S \leftarrow \{j : 1 \leq j < i \wedge Y_j || M_j = Y_i || M_i\}$
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 σ_i

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

PSign(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 σ_i

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

PSign(M) // G_7
$i \leftarrow i+1$; $M_i \leftarrow M$
Return σ_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 $(j \leq 0 \vee j > i)$ 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 κ_j

Open(j) // G_6
If $(j \leq 0 \vee j > i)$ 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 κ_j

Open(j) // G_7
If $(j \leq 0 \vee j > i)$ 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 κ_j

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

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

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

$\mathbf{H}(x)$ // 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 12: Game sequence used in proof of Theorem 4.2.

defined as the probability that $J \geq 1$ in the experiment

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

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 ρ 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

$$\mathrm{frk} \;\geq\; \mathrm{acc} \cdot \left(\frac{\mathrm{acc}}{q} - \frac{1}{h} \right) \quad \text{and} \quad \mathrm{acc} \;\leq\; \frac{q}{h} + \sqrt{q \cdot \mathrm{frk}}\, . \tag{18}$$

Proof of Part 1.: Let $q = q_s + q_H$ and consider games $G_0 - G_7$ of Figure 12. We have

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F) \;&=\; \Pr\left[\, G_0^F \,\right] \;=\; \Pr\left[\, G_1^F \,\right] \;=\; \Pr\left[\, G_2^F \,\right] \\
&=\; \Pr\left[\, G_3^F \,\right] + \Pr\left[\, G_2^F \,\right] - \Pr\left[\, G_3^F \,\right] \;\leq\; \Pr\left[\, G_3^F \,\right] + \Pr\left[\, \mathsf{BD}_3 \,\right] \\
\Pr\left[\, G_3^F \,\right] \;&=\; \Pr\left[\, G_4^F \,\right] \;=\; \Pr\left[\, G_5^F \,\right] + \Pr\left[\, G_4^F \,\right] - \Pr\left[\, G_5^F \,\right] \\
&\leq\; \Pr\left[\, G_5^F \,\right] + \Pr\left[\, \mathsf{BD}_5 \,\right] \;\leq\; \Pr\left[\, G_6^F \,\right] + \Pr\left[\, \mathsf{BD}_6 \,\right] . \\
\Pr\left[\, \mathsf{BD}_3 \,\right] \;&\leq\; \sum_{i=1}^{i-1} \left(\frac{i-1}{p} + \frac{q_H + q_o}{p} \right) \;\leq\; \frac{q_s^2 + 2q_s(q_H + q_o)}{2p} . \\
\Pr\left[\, \mathsf{BD}_5 \,\right] \;&\leq\; \frac{q_s q_H}{p} .
\end{aligned}$$

So

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

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 **Initialize** of G_7. It answers F's queries to **PSign**, **H**, **Open** in the same way as G_7. When F outputs $(M, (\sigma, \kappa))$, algorithm A lets

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

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 **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 D.1 and let frk be defined as there. Now 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 $\mathrm{Ind}(X \| Y \| M) = I$ and $\mathrm{Ind}(X \| Y' \| M') = I'$. Now in the first execution of A it must be that

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 $\mathit{bad} \leftarrow \texttt{true}$;
$\quad \boxed{\sigma \leftarrow H[X_b \| Y \| M]}$
$H[X_b \| Y \| M] \leftarrow \sigma$
Return σ

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

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

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

Figure 13: Game sequence used in proof of Theorem 4.3.

$H[X\|Y\|M]$ was defined by an H-query of F rather than by **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 **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. Now by Lemma D.1 and the above

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

Part **1.** of the theorem follows. ▮

Proof of Part 2.: We use games G_0, G_1, G_2 of Figure 13. We have

$$\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(A) = 2 \cdot \Pr\left[\, G_0^A \,\right] - 1 \,. \tag{19}$$

Since games G_0 and G_1 are equivalent, we have

$$\Pr\left[\, G_0^A \,\right] = \Pr\left[\, G_1^A \,\right] . \tag{20}$$

Games G_1 and G_2 are identical until *bad*. Then based on Lemma 3.1, we have

$$\Pr\left[\, G_1^A \,\right] \;=\; \Pr\left[\, G_1^A \,\right] - \Pr\left[\, G_2^A \,\right] + \Pr\left[\, G_2^A \,\right] \;\leq\; \Pr\left[\, \mathsf{BD}_2 \,\right] + \Pr\left[\, G_2^A \,\right] . \tag{21}$$

Combining (19), (20) and (21), we get

$$\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(A) \leq 2 \cdot (\Pr\left[\, G_2^A \,\right] + \Pr\left[\, \mathsf{BD}_2 \,\right]) - 1 \,. \tag{22}$$

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

$$\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(A) \leq 2 \cdot \Pr\left[\, \mathsf{BD}_2 \,\right]. \tag{23}$$

Initialize // H_0, H_1
$b \leftarrow\$ \{0,1\}$
$(vk_0, sk_0) \leftarrow\$ \mathsf{PKG}()$
$(vk_1, sk_1) \leftarrow\$ \mathsf{PKG}()$

LR(M_0, M_1) // $\boxed{H_0}$, H_1
For $i = 1$ to n
  $(\sigma_i, \kappa_i) \leftarrow \mathsf{PSIG}(sk_{M_b[i]}, i)$
$\sigma \leftarrow (0, \sigma_1 \| \ldots \| \sigma_n \| vk_0 \| vk_1)$
If $vk_0 = vk_1$ then $bad \leftarrow$ `true`; $\boxed{\sigma \leftarrow (1, M_b)}$
Return σ

Finalize(d) // H_0,H_1
Return $d = b$

Initialize // $G_j, L_j (0 \leq j \leq n)$
$(vk_0, sk_0) \leftarrow\$ \mathsf{PKG}()$
$(vk_1, sk_1) \leftarrow\$ \mathsf{PKG}()$

LR(M_0, M_1) // $\boxed{G_j}$,$L_j (0 \leq j \leq n)$
If $(M_0[j] = 1 \wedge M_1[j] = 0)$ then
  $\boxed{(vk, sk) \leftarrow (vk_0, sk_0)}$
  $\boxed{(vk_0, sk_0) \leftarrow (vk_1, sk_1)}$
  $\boxed{(vk_1, sk_1) \leftarrow (vk, sk)}$
For $i = 1, \ldots, j$ do $(\sigma_i, \kappa_i) \leftarrow\$ \mathsf{PSIG}(sk_{M_1[i]}, i)$
For $i = j+1, \ldots, n$ do $(\sigma_i, \kappa_i) \leftarrow\$ \mathsf{PSIG}(sk_{M_0[i]}, i)$
$\sigma \leftarrow (0, \sigma_1 \| \ldots \| \sigma_n \| vk_0 \| vk_1)$
Return σ

Finalize(d) // $G_j, L_j (0 \leq j \leq n)$
Return $d = 1$

Figure 14: Game sequence used in proof of Theorem 5.2.

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

$$\Pr[\,\mathsf{BD}_2\,] \leq \frac{q_H}{p}\ . \tag{24}$$

Part **2.** of Theorem 4.3 follows from (23) and (24). ▮

Proof of Part 3.: 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 $\mathrm{UNAMB}_{\mathcal{PS}}$, then it must be that $X_0, X_1 \in G$ and $|\sigma| = k$ and $\kappa_0, \kappa_1 \in \mathbb{Z}_p$ and $\mathbf{H}(vk_0 \| Y_0 \| M_0) = \mathbf{H}(vk_1 \| Y_1 \| M_1) = \sigma$ 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 $q_H^2 / 2^{k+1}$. ▮

E Proof of Theorem 5.2

Proof: Consider games H_0, H_1 in Figure 14. We have

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

H_1 and H_0 are identical until bad. By Lemma 3.1, we have

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

Lemma 5.1 gives us F such that

$$\Pr[\,\mathsf{BD}_1\,] \leq \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(F)\ .$$

It remains to design B so that

$$2(\Pr\left[\,H_1^A\,\right]-1) \le n \cdot \mathbf{Adv}^{\mathrm{uf}}_{\mathcal{PS}}(B)\ . \tag{25}$$

Towards this end consider games $G_j, L_j (0 \le j \le n)$ of Figure 14. It is easy to see

$$2\Pr\left[\,H_1^A\,\right]-1 = \Pr\left[\,L_n^A\,\right] - \Pr\left[\,L_0^A\,\right]\ . \tag{26}$$

The boxed code included in G_j is the key-swap that swaps the roles of $(vk_0, sk_0), (vk_1, sk_1)$ under certain conditions. However since $(vk_0, sk_0), (vk_1, sk_1)$ are independently chosen and only seen by A through the response to the **LR** query, swapping them has no effect visible to A, meaning

$$\Pr\left[\,G_j^A\,\right] = \Pr\left[\,L_j^A\,\right](1 \le j \le n)\ . \tag{27}$$

We will design B so that

$$\mathbf{Adv}^{\mathrm{an}}_{\mathcal{PS}}(B) = \frac{1}{n}(\Pr\left[\,G_n^A\,\right] - \Pr\left[\,G_0^A\,\right])\ . \tag{28}$$

Putiing together (26), (27) and (28) yields (25)and completes the proof.

Adversary B gets input $(vk_0, sk_0), (vk_1, sk_1)$. It picks $g \leftarrow\!\!\$\ \{1, \ldots, n\}$ and then starts running A, responding to A's **LR** query via the following procedure

$\mathbf{LR}(M_0, M_1)$
If $(M_0[g] = 1 \wedge M_1[g] = 0)$ then
  $(vk, sk) \leftarrow (vk_0, sk_0)$; $(vk_0, sk_0) \leftarrow (vk_1, sk_1)$
  $(vk_1, sk_1) \leftarrow (vk, sk)$
For $i = 1, \ldots, g-1$ do $(\sigma_i, \kappa_i) \leftarrow\!\!\$\ \mathsf{PSIG}(sk_{M_1[i]}, i)$
If $(M_0[g] = M_1[g])$ then $(\sigma_g, \kappa_g) \leftarrow\!\!\$\ \mathsf{PSIG}(sk_{M_1[g]}, g)$
Else $(\sigma_g, \kappa_g) \leftarrow\!\!\$\ \mathbf{CH}(g)$
For $i = g+1, \ldots, n$ do $(\sigma_i, \kappa_i) \leftarrow\!\!\$\ \mathsf{PSIG}(sk_{M_0[i]}, i)$
$\sigma \leftarrow (0, \sigma_1 \| \ldots \| \sigma_n \| vk_0 \| vk_1)$
Return σ

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

$$\Pr\left[\,d = 1 \mid g = j \wedge b = 1\,\right] = \Pr\left[\,G_j^A\,\right](1 \le j \le n)\ . \tag{29}$$

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

$$\Pr\left[\,d = 1 \mid g = j \wedge b = 0\,\right] = \Pr\left[\,G_{j-1}^A\,\right](1 \le j \le n)\ . \tag{30}$$

To justify this consider two cases. First, if $M_0[j] = M_1[j]$ then the code in B's simulated **LR** oracle is equivalent to the one in G_{j-1} in this same case. Second, if $M_0[j] \ne M_1[j]$, let $c = M_0[j]$. Then (σ_j, κ_j) is produced by $\mathbf{CH}(j)$ under vk_c. (we use here that the key swap occurs if $c = 1$.) But $vk_c = vk_{M_0[j]}$, since $c = M_0[j]$, so this corresponds to G_{j-1}. Now from (29) and (30) we have

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

which yields (28) as desired. ∎

F Proof of Theorem 5.3

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 $\sigma_1 \| \ldots \| \sigma_n \| vk_0 \| vk_1$ where $|\sigma_i| = l$, the latter being the length of a signature in $\mathcal{PS}$. 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 $vk_0 \neq vk_1$. 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 $(vk_0, vk_1, j, j, \sigma_j, \kappa_{0,j}, \kappa_{1,j})$. ∎