

A preliminary version of this paper appears in *Proceedings of the 10th International Conference on Theory and Practice of Public-Key Cryptography - PKC 2007*, Lecture Notes in Computer Science Vol. 4450, pp. 201–216, T. Okamoto and X. Wang eds., Springer-Verlag, 2007. This full version will appear in the IET Information Security Journal under the title “Two-Tier Signatures from the Fiat-Shamir Transform, with Applications to Strongly Unforgeable and One-Time Signatures.”

# Two-Tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir without Random Oracles

Mihir Bellare \*

Sarah Shoup †

## Abstract

We show how the Fiat-Shamir transform can be used to convert three-move identification protocols into *two-tier signature schemes* (a primitive we define) with a proof of security that makes a standard assumption on the hash function rather than modeling it as a random oracle. The result requires security of the starting protocol against *concurrent* attacks. We can show that numerous protocols have the required properties and so obtain numerous efficient two-tier schemes. Our first application is an efficient transform of any unforgeable signature scheme into a strongly unforgeable one, which uses as a tool any two-tier scheme. (This extends work of Boneh, Shen and Waters whose transform only applies to a limited class of schemes.) The second application is new one-time signature schemes that, compared to one-way function based ones of the same computational cost, have smaller key and signature sizes.

## 1 Introduction

Recall that the Fiat-Shamir (FS) transform [FS87] is a way to obtain a signature scheme from a three-move identification protocol by “collapsing” the interaction via a hash function. (Briefly, the signature consists of the two prover moves corresponding to a verifier challenge set to be the hash of the first prover move and the message being signed.) There are lots of protocols to which the transform can be applied, and the resulting signature schemes include some of the most efficient known (eg. [Sch91, GQ90, GQ88]). Furthermore, due to their algebraic properties, FS-transform-derived signature schemes lend themselves nicely to extensions such as to blind [PS00], multi [MOR01, BN06] or group [BBS04] signatures to name just a few. For these reasons, the transform is popular and widely used.

Naturally, one would like that the constructed signature scheme meets the standard notion of unforgeability under chosen-message attack (uf-cma) of [GMR88]. Results of [PS00, OO98, AABN02] say this is true in the random oracle (RO) model (meaning, if the hash function is a random oracle) as long as the starting protocol is itself secure (we will discuss in what sense later). However, Goldwasser and Tauman-Kalai [GK03]

---

\*Dept. of Computer Science and Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093-0404, USA. Email: [mihir@cs.ucsd.edu](mailto:mihir@cs.ucsd.edu). URL: <http://www-cse.ucsd.edu/users/mihir>. Supported in part by NSF grant CNS-0627779 and a gift from Intel Corporation.

†Dept. of Computer Science and Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093-0404, USA. Email: [sshoup@cs.ucsd.edu](mailto:sshoup@cs.ucsd.edu). URL: <http://www-cse.ucsd.edu/users/sshoup>. Supported in part by a UCSD Powell fellowship and the above mentioned grants of the first author.

show the existence of a protocol that, under the FS transform, yields a signature scheme that is uf-cma secure when the hash function is a RO but is not uf-cma secure for any standard model implementation of the hash function. This means that the transform (at least in general) does not yield uf-cma secure schemes in the standard model.

The question we ask is whether the FS transform can, however, yield weaker-than-uf-cma but still useful types of signature schemes in the standard model. We answer this in the affirmative. We show how the FS transform yields two-tier signature schemes which are secure assuming only that the hash function is collision-resistant and the starting protocol is secure. We exhibit some applications of two-tier signatures in general and FS-derived ones in particular, namely for an efficient and general transform of uf-cma to strongly unforgeable (suf-cma) signature schemes and to implement one-time signatures that are much shorter than conventional ones of the same computational cost. Let us now look at all this in more detail.

**TWO-TIER SCHEMES.** In a two-tier scheme, a signer has a primary public key and matching primary secret key. Each time it wants to sign, it generates a fresh pair of secondary public and secret keys and produces the signature as a function of these, the primary keys and the message. Verification requires not only the primary public key but also the secondary one associated to the message. Security requires that it be computationally infeasible to forge relative to the primary public key and any secondary public key that was generated by the signer, even under a chosen-message attack.

As the reader might rightfully note, two-tier signatures are not well suited for direct signing in the standard PKI, because not just the primary but also the secondary public keys would need to be certified. However, we do not propose to use them in this direct way. Instead what we will see is that they are useful tools in building other primitives.

**BUILDING TWO-TIER SIGNATURES VIA FS.** We adapt the FS transform in a natural way to convert a three-move identification protocol into a two-tier signature scheme. (Briefly, the first prover move, rather than being in the signature, is now the secondary public key. See Section 5 for details.) We show (cf. Theorem 5.1) that the constructed two-tier scheme is secure assuming the protocol is secure (we will see exactly what this means below) and the hash function is collision-resistant. So security of FS-based two-tier signatures is guaranteed in the standard model unlike security of FS-based regular signatures which is guaranteed only in the RO model.

Both the security of regular FS-based signatures (in the RO model) and the security of our FS-based two-tier signatures (in the standard model) are based on some security assumption about the starting protocol. (Naturally, since otherwise there is no reason for the constructs to be secure.) There is, however, a difference in the two cases. Recall that security of this class of protocols can be considered under three different types of attack: passive, where the adversary merely observes interactions between the prover and honest verifier; active [FS87, FFS88], where the adversary plays a cheating verifier and engages in sequential interactions with the honest prover; or concurrent [BP02], where, as a cheating verifier, the adversary can interact concurrently with different prover clones. For (uf-cma) security of FS-based regular signatures in the RO model, it suffices that the protocol be secure against passive (i.e. eavesdropping) attack [AABN02]. Our result showing security of FS-based two-tier signatures requires however that the protocol be secure against concurrent attack. Thus, part of what makes it possible to dispense with random oracles is to start from protocols with a stronger property. However, we show that the property is in fact possessed by the bulk of example protocols, so that we lose very little in terms of actual constructions. Specifically it is easy to show appropriate security under concurrent attack for the Schnorr [Sch91], Okamoto [Oka92], and GQ [GQ90, GQ88] protocols as well as others, using techniques from the original papers and more recent analyses [BP02, BNN04]. Thereby we obtain numerous specific and efficient constructions of two-tier signatures via the FS transform.

We think this is an interesting application of concurrent security of protocols. The latter is usually motivated as being important for certain communication environments such as the Internet, while we are saying it is relevant to the security of a protocol-based signature.

**FROM UF-CMA TO SUF-CMA.** Returning again to regular (rather than two-tier) signatures, recall that strong unforgeability (suf-cma) is a stronger requirement than the usual uf-cma of [GMR88], requiring not only that the adversary can't produce a signature of a new message but also that it can't produce a new signature of an old message (i.e. one whose signature it has already obtained via its chosen-message attack). The problem we are interested in is to efficiently convert a uf-cma scheme into an suf-cma one without using random oracles.

Our work is motivated by Boneh, Shen and Waters [BSW06] who turn Waters’ uf-cma scheme [Wat05] into an suf-cma one via a transform that applies to a subclass of signature schemes that they call partitioned. Unfortunately, there seem to be hardly any schemes in this class besides Waters’, so their transform is of limited utility. We, instead, provide a general transform that applies to any uf-cma scheme. The transform uses as a tool any two-tier scheme. Instantiating the latter with an FS-based two-tier scheme we obtain efficient, standard model transforms. For example, using the Schnorr scheme, our transform confers suf-cma security while adding just one exponentiation to the signing time and increasing the signature size by only two group elements. Briefly, the idea of the transform is to have two signatures, one from the original uf-cma scheme and the other from the two-tier scheme, mutually authenticate each other. This application exploits the fact that our FS-based two-tier signatures are themselves strongly unforgeable due to properties of the starting protocols. (That is, if the adversary has seen the signature of  $m$  relative to a secondary public key, it can produce neither a different signature of  $m$  nor a signature of some  $m' \neq m$  relative to the same secondary key.)

**NEW ONE-TIME SIGNATURES.** A two-tier signature scheme yields a one-time signature scheme as a special case. (Restrict to a single secondary key.) Thus we obtain FS-based strongly unforgeable one-time signatures. These turn out to be interesting because they have smaller key and signature sizes than conventional one-way function based one-time schemes of the same computational cost. Specifically, say we are signing a 160-bit message (which is the hash of the real message). Our Schnorr-instantiated FS-based one-time scheme implemented over a 160-bit elliptic curve group has key size 480 bits, signature size 160 bits, key-generation time two exponentiations, signing time one multiplication, and verifying time one exponentiation. Let us contrast this with what is achieved by the best one-way function based one-time signature schemes, namely those of [EGM96, BM96]. Unforgeability is proved by [EGM96] under the assumption that the one-way function is quasi-one-way. We observe that the scheme is strongly unforgeable under the additional assumption that the function is collision-resistant. So, let us use SHA-1 as the one-way function. The resulting schemes exhibit the following size to computation tradeoff. For any positive integer  $t$  dividing 160, there is a one time scheme with key and signature size  $(1 + 160/t) \cdot 160$  and key-generation, signing and verifying time  $(160/t) \times 2^t$  hash computations. An implementation with the crypto++ library [Dai] indicates that an exponentiation in a 160-bit group costs about 3,300 hashes. To match the key-generation time of two exponentiations (which is the largest of the computation times in our algebraic scheme) we thus want to choose  $t$  such that  $(160/t) \times 2^t \approx 6,600$ . This yields  $t \approx 8.44$ , but this is not a valid value (recall  $t$  must divide 160), so we (generously) set  $t = 10$ , yielding the shortest possible signature for key-generation time costing at least two exponentiations. The key and signature sizes are now each 2,720 bits, which is much more than in our scheme. (And at this point, while key-generation time in the one-way function scheme is about 2.5 as much as in our scheme, signing time is 16368 hashes, much more than the one multiplication and hash required to sign in the Schnorr scheme.) Note we would get the same efficiency gains using the standard Schnorr [Sch91] signature scheme instead of our scheme, but the proof of the former uses random oracles [PS00].

Our new one-time signature scheme is interesting for applications like the DDN and Lindell constructions of IND-CCA public-key encryption schemes [DDN00, Lin06], the IBE-based constructions of IND-CCA schemes of [CHK06], and the composition of encryption schemes [DK05]. All of these make use of strongly unforgeable one-time signatures, and the reduced key size of the latter results in reduced ciphertext size for the encryption schemes they build.

**ALTERNATIVE TWO-TIER SCHEMES AND THEIR IMPLICATIONS.** We noted above that two-tier schemes yield (strongly unforgeable) one-time ones as a special case. Conversely, one can also construct a two-tier scheme from any strongly unforgeable one-time scheme. (Set the primary keys to empty, and use a new instance of the one-time scheme for each secondary key. See Section 6 for details on both these transforms.) However, FS-based two tier schemes have smaller key and signature sizes than two-tier schemes of the same computational cost built from any known strongly unforgeable one-time schemes.

We could have based our transform (of uf-cma schemes into suf-cma ones) on strongly unforgeable one-time schemes rather than on two-tier schemes, and we could have built FS-based strongly unforgeable one-time schemes directly rather than first building two-tier schemes. Two-tier schemes however have the advantage over using one-time schemes that any key information that is long-lived across multiple instances of a one-time scheme can be re-used, resulting in shorter keys. This results in shorter signatures for the

suf-cma schemes built by our transform. Overall it seemed simple and worthwhile enough to make the optimization (meaning to introduce and use two-tier signatures) and hence we have done so.

RELATED WORK. Cramer and Damgård [CD95] present a non-RO transform of protocols with certain properties into signature schemes. Their transform is not the FS one (it is more complex and less efficient) but they obtain regular uf-cma signature schemes while we obtain only two-tier schemes.

Independently of our work, others have extended [BSW06] to provide general transforms of unforgeable signature schemes into strongly unforgeable ones. The transform of Huang, Wong and Zhao [HWZ06] is similar to the special case of ours with a two-tier signature scheme built from a strongly unforgeable one-time signature scheme as per Theorem 6.2. However, this yields large signatures. Teranishi, Oyama and Ogata [TOO06] present a discrete log, chameleon commitment based transform. Steinfeld, Pieprzyk, and Wang [SPW07] present a general transform which uses two randomized trapdoor hash functions.

We clarify that, above, when we discuss transforms (of uf-cma signature schemes into suf-cma ones), we mean efficient transforms. A one-way function based, polynomial time but not efficient transform of uf-cma signature schemes (in fact, one-time uf-cma signature schemes) into suf-cma signature schemes was already provided by Goldreich [Gol04].

Let us call a signature scheme kr-cma secure if it is computationally infeasible to recover the secret key under a chosen-message attack. Interestingly, the FS transform can yield signature schemes that are kr-cma secure in the standard model. Specifically, Paillier and Vergnaud [PV05] show that the Schnorr signature scheme is kr-cma secure under the one-more discrete logarithm assumption of [BNPS03] assuming only that the hash function is collision resistant.

An online-offline signature scheme [EGM96, ST01] is a standard signature scheme in which the signing process is separated into two parts for the sake of efficiency. In the offline phase, before seeing the message, the signer computes some information  $ssk$  and the secret key. Although in general the notion of online/offline signatures is unrelated to two-tier signatures, the specific schemes of [EGM96, ST01] do bear some resemblance to our two-tier approach.

## 2 Definitions

NOTATION AND CONVENTIONS. We denote by  $a_1 \| \dots \| a_n$  a string encoding of  $a_1, \dots, a_n$  from which the constituent objects are uniquely recoverable. We denote the empty string by  $\varepsilon$ . Unless otherwise indicated, an algorithm may be randomized. A collision for a function  $h$  is a pair  $x, y$  of distinct points in its domain such that  $h(x) = h(y)$ . If  $A$  is a randomized algorithm then  $y \stackrel{\$}{\leftarrow} A(x_1, \dots)$  denotes the operation of running  $A$  with fresh coins on inputs  $x_1, \dots$  and letting  $y$  denote the output. If  $S$  is a (finite) set then  $s \stackrel{\$}{\leftarrow} S$  denotes the operation of picking  $s$  uniformly at random from  $S$ . If  $X = x_1 \| x_2 \| \dots \| x_n$ , then  $x_1 \| x_2 \| \dots \| x_n \leftarrow X$  denotes the operation of parsing  $X$  into its constituents. Similarly, if  $X = (x_1, x_2, \dots, x_n)$  is an  $n$ -tuple, then  $(x_1, x_2, \dots, x_n) \leftarrow X$  denotes the operation of parsing  $X$  into its elements.

For simplicity our security definitions are non-asymptotic, meaning we associate an advantage to an adversary and formal results simply relate advantages. Having an explicit security parameter is just a notational extension.

SIGNATURES. A (digital) signature scheme  $DS = (KG, SGN, VF)$  is specified as usual by three algorithms. Via  $(PK, SK) \stackrel{\$}{\leftarrow} KG$  a prospective signer can generate its public and associated secret key. Via  $\sigma \stackrel{\$}{\leftarrow} SGN(SK, M)$  the signer can produce a signature  $\sigma$  on a message  $M \in \{0, 1\}^*$ . Via  $d \leftarrow VF(PK, M, \sigma)$ , a verifier can run the deterministic verification algorithm to get a decision bit  $d \in \{0, 1\}$ . We require perfect consistency, meaning that

$$\Pr \left[ (PK, SK) \stackrel{\$}{\leftarrow} KG ; \sigma \stackrel{\$}{\leftarrow} SGN(SK, M) : VF(PK, M, \sigma) = 1 \right] = 1$$

for all messages  $M$ . To define security consider the following game involving an adversary  $A$ :

$$(PK, SK) \stackrel{\$}{\leftarrow} KG ; (M, \sigma) \stackrel{\$}{\leftarrow} A^{SGN(SK, \cdot)}(PK) .$$

The adversary is given a signing oracle  $SGN(SK, \cdot)$  and the public key, and outputs a message and candidate signature. Let  $M_1, \dots, M_q$  denote the messages queried by  $A$  to its oracle in its chosen-message attack, and let  $\sigma_1, \dots, \sigma_q$  denote the signatures returned by the oracle, respectively. We say that  $A$  forges

<b>Oracle</b> SPKO() $i \leftarrow i + 1$ $(spk_i, ssk_i) \stackrel{\$}{\leftarrow} \text{skg}(ppk, psk)$ Return $spk_i$	<b>Oracle</b> SIGNO( $j, m$ ) If $j > i$ OR $j \in U$ then return $\perp$ $U \leftarrow U \cup \{j\}$ $s \stackrel{\$}{\leftarrow} \text{sgn}(psk, ssk_j, m)$ Return $s$
---	--

Figure 1: Oracles for adversary attacking two-tier scheme  $\text{ds} = (\text{pkg}, \text{skg}, \text{sgn}, \text{vf})$ .

if  $\text{VF}(PK, M, \sigma) = 1$  and  $M \notin \{M_1, \dots, M_q\}$ . We say that  $A$  strongly forges if  $\text{VF}(PK, M, \sigma) = 1$  and  $(M, \sigma) \notin \{(M_1, \sigma_1), \dots, (M_q, \sigma_q)\}$ . We let  $\text{Adv}_{\text{DS}}^{\text{uf-cma}}(A)$  and  $\text{Adv}_{\text{DS}}^{\text{suf-cma}}(A)$  denote, respectively, the probability that  $A$  forges and the probability that it strongly forges. The first measure represents the standard uf-cma notion of [GMR88], while the second represents strong unforgeability (suf-cma).

Recall that a one-time signature scheme is simply one where security (of whatever type) is required only with respect to adversaries that make at most one query in their chosen-message attack.

### 3 Two-tier signatures

**SYNTAX.** A two-tier signature scheme  $\text{ds} = (\text{pkg}, \text{skg}, \text{sgn}, \text{vf})$  is specified by four algorithms. They are called the primary key-generation, secondary key-generation, signing and verifying algorithms, respectively, and the last is deterministic. Via  $(ppk, psk) \stackrel{\$}{\leftarrow} \text{pkg}$ , a prospective signer generates a primary public key  $ppk$  and associated primary secret key  $psk$ . Think of these as the keys at the first tier of the two-tier scheme. The signer may then at any time generate a secondary public key  $spk$  and associated secondary secret key  $ssk$  via  $(spk, ssk) \stackrel{\$}{\leftarrow} \text{skg}(ppk, psk)$ . These will be the second tier keys, and there can be many of them. Via  $s \stackrel{\$}{\leftarrow} \text{sgn}(psk, ssk, m)$  the signer can generate a signature of a message  $m$ . Via  $d \leftarrow \text{vf}(ppk, spk, m, s)$ , a verifier can produce a decision bit  $d \in \{0, 1\}$  indicating whether or not  $s$  is a valid signature of  $m$  relative to  $ppk, spk$ . We require perfect consistency, meaning that

$$\Pr \left[ (ppk, psk) \stackrel{\$}{\leftarrow} \text{pkg} ; (spk, ssk) \stackrel{\$}{\leftarrow} \text{skg}(ppk, psk) ; s \stackrel{\$}{\leftarrow} \text{sgn}(psk, ssk, m) : \text{vf}(ppk, spk, m, s) = 1 \right] = 1$$

for all messages  $m$ .

In usage, a signer will have a single primary key pair. It will generate and use a fresh secondary key pair for each message so that the secondary key pairs are one-time. Since generation of a secondary key pair does not require knowing the message, this generation can either be done when the message to be signed arrives, or off-line, in advance.

**SECURITY.** To define security, consider the following game. We let  $(ppk, psk) \stackrel{\$}{\leftarrow} \text{pkg}$ , initialize a set  $U$  to  $\emptyset$  and initialize a counter  $i$  to 0. We then run an adversary  $A$  on input  $ppk$  with access to the oracles shown in Figure 1.  $A$  can obtain a fresh secondary public key at any time by calling its secondary public-key oracle SPKO.  $A$  can obtain a signature of a message  $m$  of its choice under an already generated secondary public key  $spk_j$  by calling the signing oracle SIGNO on inputs  $j, m$ , where  $j \geq 1$ . However,  $A$  cannot obtain more than one signature under a particular secondary public key. (This restriction is enforced by the oracle via the set  $U$ .) Finally  $A$  outputs a forgery, which must be a triple of the form  $(l, m, s)$ . Let  $(j_1, m_1), \dots, (j_q, m_q)$  denote the queries made by  $A$  to its SIGNO oracle in its chosen-message attack, and let  $s_1, \dots, s_q$  denote the signatures returned by the oracle, respectively. We say that  $A$  wins if  $\text{vf}(ppk, spk_l, m, s) = 1$  and  $1 \leq l \leq i$  but  $(l, m, s) \notin \{(j_1, m_1, s_1), \dots, (j_q, m_q, s_q)\}$ . Here  $i$  is the final value of the counter, meaning the number of queries  $A$  made to SPKO. The probability that  $A$  wins is denoted  $\text{Adv}_{\text{ds}}^{\text{suf-cma}}(A)$ .

Notice that this definition is of strong unforgeability, meaning this has been built in as a requirement. We do this because it is what the applications need and also what the FS-based constructs naturally provide.

**DISCUSSION.** Two-tier schemes are hybrids of regular and one-time schemes. If the secondary keys are empty, we have a regular scheme. If the primary keys are empty, we have multiple instances of a one-time scheme. See Section 6 for more on relations to one-time schemes.

ONLINE TWO-TIER SIGNATURES. We say that our adversary  $A$  is online if, whenever it makes a query  $j, m$  to SIGNO, it is the case that  $j = i$ . Online adversaries are a subclass of the adversaries considered above, and we say a two-tier scheme has online security if it is secure in the above sense with respect to this subclass of adversaries.

We remark that while security of our FS-based two-tier schemes requires security of the identification protocol under concurrent attack, online security of the same two-tier scheme only requires security of the identification protocol under active attack. Furthermore, online security of the two-tier scheme is sufficient for both the applications of this paper, namely Theorem 4.1 and Theorem 6.1.

The reason we have nonetheless considered full (rather than online) security is that there may be settings in which the signer wishes to pre-compute secondary keys offline, in which case our definition guarantees security even if the secondary public keys become exposed to the adversary. Meanwhile, the identification protocols we consider are secure under concurrent attack anyway—that is, there is no cost in computation or assumption to provide security under concurrent as opposed to active attack—so we do not pay a price for the extra security.

## 4 From uf-cma to suf-cma

Suppose we are given a uf-cma signature scheme  $DS$  and want to transform it into an suf-cma signature scheme  $\overline{DS}$ , efficiently and without random oracles. This problem was recently considered by [BSW06] who provided a transform that works under the assumption that the starting uf-cma scheme is what they call “partitioned.” However, there are few examples of partitioned schemes. In this section, we provide a general transform, namely one that applies to any starting uf-cma scheme. It uses an arbitrary two-tier scheme as an auxiliary tool. The transform does not use random oracles, and, when instantiated with our Schnorr FS-based two-tier schemes, matches that of [BSW06] in computational overhead while providing signatures that are longer by only one group element.

IDEA. The problem is that, given a signature  $S$  of a message  $M$  under a public key  $PK$  of the uf-cma scheme, it may be possible for an adversary to transform it into a different signature  $S' \neq S$  of  $M$  under  $PK$ . A natural idea to prevent this is to have the signer sign not just the message but the signature, namely the signature of  $M$  is a pair  $(S, s)$  where  $s$  is a signature of  $S$ , say under a different public key. Now, if the adversary wants to use  $S'$  in a forgery, it must also produce a signature  $s'$  of  $S'$  and it is not clear how to do this if the scheme is uf-cma. The reason this does not immediately work is that the problem has simply been moved onto the second signature, for the adversary may be able to leave  $S$  unchanged and modify  $s$  into a different signature  $s'$  of  $S$ . To prevent this, it would suffice that the second signature is produced under an suf-cma scheme, but this does not help since we do not have such a scheme. (We are trying to build one.) Instead, we will use a (strongly unforgeable) two-tier scheme. (And then build such schemes in Section 5.) The difficulty with using this weaker form of signature, however, is that security only holds relative to a secondary public key that the adversary is presumed not to be able to manipulate, and each signature needs a new secondary key. What we do is produce a new secondary public key for each signature, authenticate this key and the message with the uf-cma scheme, and then authenticate the resulting signature with the two-tier scheme under the same secondary key. In this way, the two signatures essentially mutually authenticate each other, which is the way out of the cyclicity that came up above.

DETAILS. We now detail the construction and prove its security. We begin by describing the transform. Let  $DS = (\text{KG}, \text{SGN}, \text{VF})$  be the given uf-cma scheme. Let  $ds = (\text{pkg}, \text{skg}, \text{sgn}, \text{vf})$  be a (any) given two-tier scheme. We associate to these the signature scheme  $\overline{DS} = (\overline{\text{KG}}, \overline{\text{SGN}}, \overline{\text{VF}})$  defined as follows. The key-generation algorithm  $\overline{\text{KG}}$  runs  $\text{KG}$  to get  $(PK, SK)$ , runs  $\text{pkg}$  to get  $(ppk, psk)$ , and returns  $\overline{PK} = PK || ppk$  as the public key and  $\overline{SK} = SK || psk$  as the secret key. The new signing and verifying algorithms appear in Figure 2.

Putting the figure into words, to sign message  $\overline{M}$ , first generate a fresh key pair  $(spk, ssk)$  for the two-tier signature scheme. View  $spk || \overline{M}$  as a message  $M$  and sign it using the given signature scheme under secret key  $SK$  to get a signature  $S$ . Now view  $S$  as a message and sign it using the two-tier signature scheme under secret key  $ssk$ . The signature  $\overline{S}$  of  $\overline{M}$  consists of  $S, spk$  and  $s$ . The following implies that the constructed scheme  $\overline{DS}$  is strongly unforgeable if  $DS$  is unforgeable and the two-tier scheme  $ds$  is strongly unforgeable.

<b>Algorithm</b> $\overline{\text{SGN}}(\overline{SK}, \overline{M})$ Parse $\overline{SK}$ as $SK  psk$ $(spk, ssk) \xleftarrow{s} \text{skg}(ppk, psk)$ $M \leftarrow spk  \overline{M}$ $S \xleftarrow{s} \text{SGN}(SK, M)$ $s \xleftarrow{s} \text{sgn}(psk, ssk, S)$ $\overline{S} \leftarrow S  spk  s$ Return $\overline{S}$	<b>Algorithm</b> $\overline{\text{VF}}(\overline{PK}, \overline{M}, \overline{S})$ Parse $\overline{PK}$ as $PK  ppk$ Parse $\overline{S}$ as $S  spk  s$ $M \leftarrow spk  \overline{M}$ If $\text{VF}(PK, M, S) = 0$ then return 0 If $\text{vf}(ppk, spk, S, s) = 0$ then return 0 Return 1
---	---

Figure 2: Algorithms for the transform from uf-cma to suf-cma.

**Theorem 4.1** Let  $\overline{\text{DS}}$  be the signature scheme associated to signature scheme DS and two-tier signature scheme ds as described above. Let  $\overline{F}$  be an adversary attacking the strong unforgeability of  $\overline{\text{DS}}$  and making at most  $q$  signing queries. Then there exist adversaries  $F, f$  attacking the unforgeability of DS and the strong unforgeability of ds, respectively, such that

$$\mathbf{Adv}_{\overline{\text{DS}}}^{\text{suf-cma}}(\overline{F}) \leq \mathbf{Adv}_{\text{DS}}^{\text{uf-cma}}(F) + \mathbf{Adv}_{\text{ds}}^{\text{suf-cma}}(f).$$

Furthermore  $F$  and  $f$  make at most  $q$  signing queries, and they each run in the time of  $\overline{F}$  plus an overhead that is linear in  $q$ . ■

**Proof of Theorem 4.1:** Adversaries  $F, f$  are described in Figure 3. Explanations and the analysis follow.

Given public key  $PK$ , adversary  $F$  generates its own  $(ppk, psk)$  pair for the two-tier signature scheme and provides  $\overline{F}$  with the public key  $\overline{PK}$ , where  $\overline{PK} = pk||ppk$ . It replies to  $\overline{F}$ 's signing queries as in the description of the scheme, using its own SGN oracle to produce signatures under the secret key  $SK$  associated to  $PK$ . This provides a perfect simulation of the view of  $\overline{F}$  in its game. If  $\overline{F}$  produces a valid forgery  $(\overline{M}, \overline{S})$ , where  $\overline{S} = S||spk||s$ , it follows from the  $\overline{\text{VF}}$  algorithm that  $\text{VF}(PK, spk||\overline{M}, S) = 1$ . Thus  $(spk||\overline{M}, S)$  is a valid forgery for  $F$  if  $spk||\overline{M}$  was never previously queried to the signing oracle. If we let **Bad** be the event that  $spk||\overline{M}$  was previously queried to the signing oracle, it follows that

$$\mathbf{Adv}_{\overline{\text{DS}}}^{\text{uf-cma}}(F) \geq \Pr[\overline{F} \text{ strongly forges} \wedge \overline{\text{Bad}}].$$

Given primary public key  $ppk$ , adversary  $f$  generates its own  $(PK, SK)$  pair for the DS signature scheme and provides  $\overline{F}$  with the public key  $\overline{PK} = PK||ppk$ . It replies to  $\overline{F}$ 's signing queries as in the description of the scheme, but using its own SPKO and SIGNO oracles to generate signatures under ds. This provides a perfect simulation of the view of  $\overline{F}$  in its game. If  $\overline{F}$  produces a valid forgery  $(\overline{M}, \overline{S})$ , where  $\overline{S} = S||spk||s$ , it follows from the  $\overline{\text{VF}}$  algorithm that  $\text{vf}(ppk, spk, S, s) = 1$ . Adversary  $f$  then searches for a  $j$  such that  $spk_j||\overline{M}_j = spk||\overline{M}$ . Such a  $j$  exists when the event **Bad** occurs. In this case,  $(j, S, s)$  is a valid forgery for adversary  $f$  if  $f$  did not previously query  $(j, S)$  to its oracle and receive back  $s$  as the signature. But since  $f$  queries SIGNO at index  $j$  at most once, this would mean that  $(S_j, s_j) = (S, s)$ , and thus  $(\overline{M}, \overline{S}) = (\overline{M}_j, \overline{S}_j)$ . But then  $(\overline{M}, \overline{S})$  could not be a strong forgery for adversary  $\overline{F}$ . Thus it follows that

$$\mathbf{Adv}_{\text{ds}}^{\text{suf-cma}}(f) \geq \Pr[\overline{F} \text{ strongly forges} \wedge \overline{\text{Bad}}].$$

Putting all this together we have

$$\begin{aligned} \mathbf{Adv}_{\overline{\text{DS}}}^{\text{suf-cma}}(\overline{F}) &= \Pr[\overline{F} \text{ strongly forges}] \\ &= \Pr[\overline{F} \text{ strongly forges} \wedge \overline{\text{Bad}}] + \Pr[\overline{F} \text{ strongly forges} \wedge \overline{\overline{\text{Bad}}}] \\ &\leq \mathbf{Adv}_{\text{DS}}^{\text{uf-cma}}(F) + \mathbf{Adv}_{\text{ds}}^{\text{suf-cma}}(f). \end{aligned}$$

Furthermore  $F$  and  $f$  make at most  $q$  signing queries, where  $q$  is the number of signing queries that  $\overline{F}$  makes. The overhead is the time to determine if there is a  $j$  such that  $spk_j||\overline{M}_j = spk||\overline{M}$ , which can be done in  $O(q)$  time with appropriate data structures. ■

<p><b>Adversary</b> <math>F^{\text{SGN}(SK, \cdot)}(PK)</math></p> <p><math>(ppk, psk) \stackrel{\\$}{\leftarrow} \text{pkg}</math>  <math>\overline{PK} \leftarrow PK    mpk; i \leftarrow 0</math>  Run <math>\overline{F}</math> on input <math>\overline{PK}</math>,  responding to oracle queries as follows:  On sign query <math>\overline{M}</math>:  <math>i \leftarrow i + 1; \overline{M}_i \leftarrow \overline{M}</math>  <math>(spk_i, ssk_i) \stackrel{\\$}{\leftarrow} \text{skg}(ppk, psk)</math>  <math>M_i \leftarrow spk_i    \overline{M}_i</math>  <math>S_i \stackrel{\\$}{\leftarrow} \text{SGN}(SK, M_i)</math> // using oracle  <math>s_i \stackrel{\\$}{\leftarrow} \text{sgn}(psk, ssk_i, S_i)</math>  <math>\overline{S}_i \leftarrow S_i    spk_i    s_i</math>  When <math>F</math> halts with forgery <math>(\overline{M}, \overline{S})</math>  Parse <math>\overline{S}</math> as <math>S    spk    s; M \leftarrow spk    \overline{M}</math>  If <math>\exists j (1 \leq j \leq i)</math> s.t. <math>M_j = M</math>  then abort  Else return <math>(M, S)</math></p>	<p><b>Adversary</b> <math>f^{\text{SPKO}(\cdot), \text{SIGNO}(\cdot, \cdot)}(ppk)</math></p> <p><math>(PK, SK) \stackrel{\\$}{\leftarrow} \text{KG}</math>  <math>\overline{PK} \leftarrow PK    mpk; i \leftarrow 0</math>  Run <math>\overline{F}</math> on input <math>\overline{PK}</math>,  responding to oracle queries as follows:  On sign query <math>\overline{M}</math>:  <math>i \leftarrow i + 1; \overline{M}_i \leftarrow \overline{M}</math>  <math>spk \stackrel{\\$}{\leftarrow} \text{SPKO}()</math> // using oracle  <math>M_i \leftarrow spk    \overline{M}_i</math>  <math>S_i \stackrel{\\$}{\leftarrow} \text{SGN}(SK, M_i)</math>  <math>s_i \stackrel{\\$}{\leftarrow} \text{SIGNO}(i, S_i)</math> // using oracle  <math>\overline{S}_i \leftarrow S_i    spk    s_i</math>  When <math>F</math> halts with forgery <math>(\overline{M}, \overline{S})</math>  Parse <math>\overline{S}</math> as <math>S    spk    s; M \leftarrow spk    \overline{M}</math>  If <math>\exists j (1 \leq j \leq i)</math> s.t. <math>M_j = M</math>  then return <math>(j, S, s)</math>  Else abort</p>
--	---

Figure 3: **Adversaries for proof of Theorem 4.1.** Adversary  $F$  is attacking uf-cma scheme DS and adversary  $f$  is attacking two-tier scheme ds.

## 5 FS-based constructions of two-tier schemes

CANONICAL IDENTIFICATION PROTOCOLS. The FS transform applies to a class of protocols we call canonical identification protocols [AABN02]. We need to have a general syntax for these protocols since the transform and its proof will refer to this. The protocol can be described as a tuple  $\text{ID} = (\mathcal{K}, P, \text{ChSet}, V)$ . Via  $(pk, sk) \stackrel{\$}{\leftarrow} \mathcal{K}$ , the (honest) prover generates its public and secret keys. Now the public key  $pk$  is viewed as an input for the verifier, while  $sk$  is a private input to the honest prover. The prover can now convince the verifier of its identity via a three move interaction as depicted in Figure 4. We refer to the moves as commitment, challenge, and response. The (honest) prover maintains a state  $\text{St}$  whose initial value is its secret key  $sk$ . In its first move, it applies  $P$  to the current conversation (which is  $\varepsilon$ ) and current state ( $\text{St} = sk$ ) to get a commitment  $\text{CM}$  and an updated state  $\text{St}$ . The former is sent to the verifier, who now draws its challenge  $\text{CH}$  at random from  $\text{ChSet}$  and sends this to the prover. The (honest) prover now lets  $\text{RP} \stackrel{\$}{\leftarrow} P(\text{CM} || \text{CH}, \text{St})$  and sends  $\text{RP}$  back to the verifier. The latter applies the deterministic function  $V$  to  $pk$  and the transcript  $\text{CM} || \text{CH} || \text{RP}$  to output the decision  $\text{DEC} \in \{0, 1\}$ . We require perfect completeness, meaning that for all  $(pk, sk)$  that can be output by  $\mathcal{K}$  we have

$$\Pr \left[ (\text{CM}, \text{St}) \stackrel{\$}{\leftarrow} P(\varepsilon, sk); \text{CH} \stackrel{\$}{\leftarrow} \text{ChSet}; \text{RP} \stackrel{\$}{\leftarrow} P(\text{CM} || \text{CH}, \text{St}) : V(pk, \text{CM} || \text{CH} || \text{RP}) = 1 \right] = 1. \quad (1)$$

Examples of canonical identification protocols include the Schnorr protocol [Sch91] illustrated in Figure 5 and the Okamoto protocol [Oka92] illustrated in Figure 6.

SECURITY NOTIONS. The “master” property of protocols in this domain is special soundness. We will consider it under different forms of attack, namely passive, active and concurrent. (We only use the last in our results but for discussions it is useful to see them all.) To define these consider the following game involving an attacker  $I$ . The game begins by picking keys via  $(pk, sk) \stackrel{\$}{\leftarrow} \mathcal{K}$ . Then there are two phases. In the first phase, adversary  $I$  gets to mount its attack on the honest prover. In a passive attack, it gets an oracle that upon being invoked (with no arguments) returns a random transcript of an interaction between the honest prover (given input  $sk$ ) and the verifier (given input  $pk$ ). In an active or concurrent attack,  $I$  gets to play the role of verifier and interact with “clones” of the honest prover. We can imagine a sequence  $P_j$  ( $j \geq 1$ ) of potential clones. Each clone maintains a state  $\text{St}_j$  and has its own random coins. The game

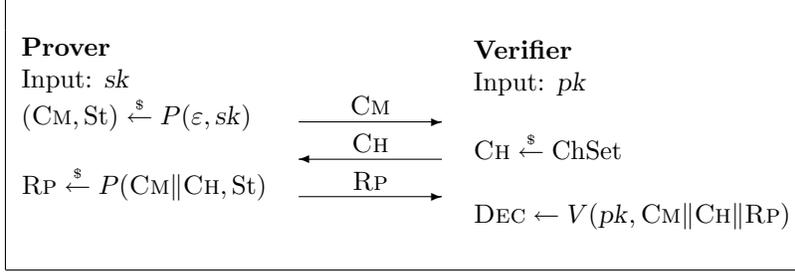


Figure 4: **Canonical Protocol**. Keys  $pk$  and  $sk$  are produced using key generation algorithm  $\mathcal{K}$ .

maintains a counter  $a$ , initially 0, and a set  $A$  of clones that are activated, initially empty. Adversary  $I$  can ask for a new clone to be activated, in which case the game increments  $a$ , computes  $(CM_a, St_a) \stackrel{s}{\leftarrow} P(\varepsilon, sk)$ , and returns  $CM_a$  to  $I$ . If the attack is concurrent, it adds  $a$  to  $A$ , but if the attack is active, it replaces  $A$  by  $\{a\}$ , meaning that only one clone can be activated at any time. If  $j \in A$  then  $I$  can send clone  $P_j$  a message  $CH_j$  representing the verifier move. Adversary  $I$  can pick this value any way it wishes, in particular not necessarily at random like the honest verifier. The game computes  $RP_j \stackrel{s}{\leftarrow} P(CM_j || CH_j, St_j)$ , returns  $RP_j$  to  $I$ , and removes  $j$  from  $A$ . (Which means no further interaction with  $P_j$  is possible.) Note that the difference between an active and concurrent attack is that in the former, the adversary is allowed to have only one clone (namely  $P_a$ ) activated at any time, corresponding to sequential interactions with the honest prover, while in a concurrent attack, any number of clones may simultaneously be activated, and  $I$  can choose a challenge sent to one of them as a function of all communications it has received from all clones so far. Note that in either case, the adversary does not see or control the internal state of a prover clone. In no case can it reset or backup a clone. After it has completed its attack (of whatever form), we enter the second phase. The adversary outputs a pair  $(CM, CH_1, RP_1), (CM, CH_2, RP_2)$  of transcripts where the commitment is the same. It wins if these transcripts are accepting but  $(CH_1, RP_1) \neq (CH_2, RP_2)$ . The probability that it wins is denoted  $\text{Adv}_{\text{ID}}^{\text{ss-atk}}(I)$ , where  $\text{atk} = \text{pa}$  if the attack is passive;  $\text{atk} = \text{aa}$  if the attack is active; and  $\text{atk} = \text{ca}$  if the attack is concurrent.

DISCUSSION. The typical formulation of special soundness is that given a pair  $(CM, CH_1, RP_1), (CM, CH_2, RP_2)$  of accepting transcripts where the commitment is the same but  $CH_1 \neq CH_2$ , one can easily find a matching secret key  $sk$ . This implies in particular that the protocol is a proof of knowledge of the secret key which in turn is crucial to proving security against impersonation under passive, active or concurrent attack. (Impersonation means that after its attack, meaning in the second phase, rather than outputting a pair of transcripts, the adversary plays the role of prover in an interaction with the honest verifier and wins if it can convince the latter to accept.) For our purposes, however, we work directly with special soundness rather than any of its derivative properties. We directly require that the probability of finding transcripts of the appropriate type is negligible rather than relating this to finding the secret key. This is similar to the security requirement used in [CD95], though they apply it to a different protocol-based transform. Note we weaken the condition under which the adversary wins from  $CH_1 \neq CH_2$  to  $(CH_1, RP_1) \neq (CH_2, RP_2)$ . We will have to prove that the resulting stronger security requirement is met by the constructs.

A  $\Sigma$  protocol [Cra96] is one that has special soundness and honest-verifier zero-knowledge. We do not explicitly require the latter as part of special soundness, although in establishing special soundness of particular protocols we might use it. Note none of the example protocols in this domain are full (i.e. even against cheating verifiers) zero-knowledge. Indeed, this is ruled out under blackbox simulation [GK96].

Special soundness is usually considered as a stand-alone property, but it is natural to consider it under the three forms of attack that exist for identification protocols as we have done.

THE TRANSFORM. We now describe how to turn a canonical identification protocol  $\text{ID} = (\mathcal{K}, P, \text{ChSet}, V)$  into a two-tier signature scheme  $\text{ds} = (\text{pkg}, \text{skg}, \text{sgn}, \text{vf})$  via the Fiat-Shamir transform. We do not use a random oracle but instead a family  $H : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \text{ChSet}$  of collision-resistant (CR) hash functions where each  $k$ -bit key  $K$  specifies a particular hash function  $H(K, \cdot)$  with domain  $\{0, 1\}^*$  and range the challenge set  $\text{ChSet}$ . (The keys will be random but public.) The primary key generation algorithm  $\text{pkg}$  lets

$K \xleftarrow{\$} \{0, 1\}^k$  and  $(pk, sk) \xleftarrow{\$} \mathcal{K}$ , and returns  $(ppk, psk) \leftarrow (K \parallel pk, K \parallel sk)$ . The  $\text{skg}$ ,  $\text{sgn}$ , and  $\text{vf}$  algorithms are as follows:

<b>Algorithm</b> $\text{skg}(ppk, psk)$ Parse $ppk$ as $K \parallel pk$ Parse $psk$ as $K \parallel sk$ $(\text{CM}, \text{St}) \xleftarrow{\$} P(\varepsilon, sk)$ $spk \leftarrow \text{CM}$ $ssk \leftarrow \text{CM} \parallel \text{St}$ Return $(spk, ssk)$	<b>Algorithm</b> $\text{sgn}(psk, ssk, m)$ Parse $psk$ as $K \parallel sk$ Parse $ssk$ as $\text{CM} \parallel \text{St}$ $\text{CH} \leftarrow H(K, \text{CM} \parallel m)$ $\text{RP} \xleftarrow{\$} P(\text{CM} \parallel \text{CH}, \text{St})$ $s \leftarrow \text{RP}$ Return $s$	<b>Algorithm</b> $\text{vf}(ppk, spk, m, s)$ Parse $ppk$ as $K \parallel pk$ $\text{CM} \leftarrow spk$ $\text{CH} \leftarrow H(K, \text{CM} \parallel m)$ $\text{RP} \leftarrow s$ $\text{DEC} \leftarrow V(pk, \text{CM} \parallel \text{CH} \parallel \text{RP})$ Return $\text{DEC}$
---	--	--

Note that in generating  $s$ , algorithm  $P$  will be executed with a challenge that, unlike the one the honest prover expects to receive, is not random but is the output of  $H(K, \cdot)$ . The implications for security are dealt with by the theorem that follows, but at this point we need to first check that it does not lead to a violation of the perfect consistency requirement of two-tier schemes. This is true because the protocol has perfect completeness as per Equation (1), which means for all values of the verifier challenge, the prover returns a response that leads the verifier to accept.

SECURITY OF THE TRANSFORM. Recall that the cr-advantage of an adversary  $F$  attacking  $H$  is

$$\text{Adv}_H^{\text{cr}}(F) = \Pr \left[ K \xleftarrow{\$} \{0, 1\}^k ; (x_1, x_2) \xleftarrow{\$} F(K) : H(K, x_1) = H(K, x_2) \wedge x_1 \neq x_2 \right].$$

The following says that if  $H$  is CR and ID is secure against concurrent attack then the two-tier scheme derived via the FS transform is secure.

**Theorem 5.1** Let  $\text{ds} = (\text{pkg}, \text{skg}, \text{sgn}, \text{vf})$  be the two-tier signature scheme associated to canonical identification protocol  $\text{ID} = (\mathcal{K}, P, \text{ChSet}, V)$  and hash function  $H: \{0, 1\}^k \times \{0, 1\}^* \rightarrow \text{ChSet}$  via the Fiat-Shamir transform as described above. Let  $f$  be an adversary attacking the strong unforgeability of  $\text{ds}$  and making at most  $q$  signing queries. Then there exists an adversary  $I$  attacking the special soundness of  $\text{ID}$  under concurrent attack, and an adversary  $F$  attacking the collision-resistance of  $H$ , such that

$$\text{Adv}_{\text{ds}}^{\text{suf-cma}}(f) \leq \text{Adv}_{\text{ID}}^{\text{ss-ca}}(I) + \text{Adv}_H^{\text{cr}}(F).$$

Furthermore  $I$  initiates at most  $q + 1$  prover clones, and the running time of each of  $I$  and  $F$  is that of  $f$  plus a constant amount of overhead. ■

**Proof of Theorem 5.1:** We assume for simplicity that if  $f$  outputs forgery  $(l, m, s)$ , it has previously issued a signing query of the form  $(l, m_l)$  for some  $m_l \neq m$ . If not, we simply transform  $f$  into an adversary  $f'$  with this property, without altering its advantage. To do so, let  $f$  behave as  $f'$  except before it outputs its forgery  $(l, m, s)$ , it queries the sign oracle with  $(l, m')$  for some  $m' \neq m$ . This increases  $q$  by at most 1.

On input public key  $pk$ , adversary  $I$  initializes a counter  $v$  to 0, a set  $V$  to empty, generates its own key  $K \xleftarrow{\$} \{0, 1\}^k$  and runs  $f$  on input  $ppk \leftarrow K \parallel pk$ .

On an SPKO query from  $f$ ,  $I$  increments  $v$  and asks its game to activate a new clone. It receives  $\text{CM}_v$  in response, sets  $spk_v \leftarrow \text{CM}_v$ , and returns this value to  $f$ . Because  $\text{CM}_v$  is generated by  $I$ 's game by running  $P(\varepsilon, sk)$ , this provides a perfect simulation for this oracle.

On a query  $\text{SIGNO}(j, m)$  from  $f$ ,  $I$  returns  $\perp$  to  $f$  if  $j > v$  or  $j \in V$ . Else it puts  $j$  in  $V$ , sets  $m_j \leftarrow m$ , and lets  $\text{CH}_j \leftarrow H(K, \text{CM}_j \parallel m_j)$ . It sends  $\text{CH}_j$  to (activated) clone  $P_j$  and receives back a response  $\text{RP}_j$ . The value  $s_j \leftarrow \text{RP}_j$  is returned to  $f$ . Because  $\text{RP}_j$  is generated by running  $P(\text{CM}_j \parallel \text{CH}_j, \text{St}_j)$ , where  $\text{St}_j$  is the state of clone  $P_j$  as maintained by  $I$ 's game, this is also a perfect simulation.

Eventually  $f$  halts with forgery  $(l, m, s)$ . Let  $\text{RP} \leftarrow s$  and  $\text{CH} \leftarrow H(K, \text{CM}_l \parallel m)$ . If this is a valid forgery, then  $V(pk, \text{CM}_l \parallel \text{CH} \parallel \text{RP}) = 1$ ,  $(m, s) \neq (m_l, s_l)$ , and  $l \leq v$ . Adversary  $I$  outputs the transcripts  $(\text{CM}_l, \text{CH}, \text{RP})$  and  $(\text{CM}_l, \text{CH}_l, \text{RP}_l)$ . Both transcripts are accepting, but  $I$  won't win if  $(\text{CH}, \text{RP}) = (\text{CH}_l, \text{RP}_l)$ . We denote this last event as  $\text{Bad}$ , and it follows that:

$$\text{Adv}_{\text{ID}}^{\text{ss-ca}}(I) \geq \Pr [f \text{ strongly forges} \wedge \overline{\text{Bad}}].$$

On input key  $K$ , CR adversary  $F$  generates its own  $pk$  and  $sk$  via  $(pk, sk) \xleftarrow{\$} \mathcal{K}$ . It sets  $(ppk, psk) \leftarrow (K \| pk, K \| sk)$  and runs  $f$  on input  $ppk$ . Since  $F$  has  $psk$ , it can act as the honest prover and respond to all of  $f$ 's queries in a straightforward way. Like  $I$ , it keeps its own counter and keeps track of all internal variables created during each query.

Eventually  $f$  halts with forgery  $(l, m, s)$ . Let  $\text{CH} \leftarrow H(K, \text{CM}_l \| m)$  and  $\text{RP} \leftarrow s$ . Adversary  $F$  will output  $(\text{CM}_l \| m, \text{CM}_l \| m_l)$  as its collision and win whenever  $H(K, \text{CM}_l \| m) = H(K, \text{CM}_l \| m_l)$  and  $m \neq m_l$ . This is always true when the event **Bad** happens and  $(l, m, s)$  is a strong forgery. Otherwise, it would hold that  $(l, m, s) = (l, m_l, s_l)$ , and thus  $f$  would not win. It follows that:

$$\mathbf{Adv}_H^{\text{cr}}(f) \geq \Pr [f \text{ strongly forges} \wedge \text{Bad}] .$$

Putting the above together we have

$$\begin{aligned} \mathbf{Adv}_{\text{ds}}^{\text{suf-cma}}(f) &= \Pr [f \text{ strongly forges}] \\ &= \Pr [f \text{ strongly forges} \wedge \text{Bad}] + \Pr [f \text{ strongly forges} \wedge \overline{\text{Bad}}] \\ &\leq \mathbf{Adv}_{\text{ID}}^{\text{ss-ca}}(I) + \mathbf{Adv}_H^{\text{cr}}(F) , \end{aligned}$$

and the theorem follows. ■

Note that our proof would have been correct if we had used  $H(K, m)$  in place of  $H(K, \text{CM} \| m)$  in the transform, as we are always dealing with pairs of transcripts with identical CM values. Though hashing the commitment with the message is not necessary, we have included it for consistency with the Fiat-Shamir transform.

The reason we needed ID to be secure under concurrent (rather than active) attack above is that the two-tier adversary is not required to sign on a secondary key immediately after acquiring it. (This models that secondary key generation may be done off-line, in advance.) So in the reduction it is necessary to have multiple clones (one per secondary key) active at once.

To instantiate the above we now seek efficient protocols for which we can prove special soundness under concurrent attack. There are actually several such protocols. We illustrate by looking at a pair of examples that are representative due to the proof techniques.

**DEFINITIONS.** In what follows,  $G$  denotes a group whose order  $p$  is a prime. (For example an appropriate elliptic curve group, or a subgroup of the group of integers modulo some prime  $q$  such that  $p$  divides  $q - 1$ .) Let  $G^* = G - \{1\}$  be the set of generators of  $G$ , where 1 is the identity element of  $G$ . We let  $\text{DLog}_g(h)$  denote the discrete logarithm of  $h \in G$  to base a generator  $g \in G^*$ . We assume  $G, p$  are fixed and known to all parties. Let

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

denote the advantage of an adversary  $A$  in attacking the discrete logarithm (dl) problem. An adversary  $A$  for the one more dl (omdl) problem [BNPS03] is given input a generator  $g \in G^*$  and has access to two oracles. The first is a challenge oracle  $\text{CHO}()$  that takes no inputs and, every time it is invoked, returns a random element of  $G$ . The second is a dl oracle  $\text{DLog}_g(\cdot)$  that, given any  $W \in G$ , returns  $\text{DLog}_g(W)$ . Let  $W_1, \dots, W_q$  denote the responses to  $A$ 's queries to its challenge oracle.  $A$ 's goal is to compute the discrete logarithms of all challenges, meaning output  $w_1, \dots, w_q \in \mathbb{Z}_p$  satisfying  $g^{w_i} = W_i$  for all  $1 \leq i \leq q$ . Of course this is easy because it has a  $\text{DLog}_g(\cdot)$  oracle. To make the task non-trivial, however, we restrict  $A$  to make strictly fewer queries to its  $\text{DLog}_G(\cdot)$  oracle than it does to its challenge oracle. Let  $\mathbf{Adv}_G^{\text{omdl}}(A)$  be the probability that  $A$  wins.

**SCHNORR IDENTIFICATION PROTOCOL.** The Schnorr identification protocol [Sch91] shown in Figure 5 is probably the most “canonical” example of a canonical identification protocol. It is secure against impersonation under passive attack under the dl assumption [Sch91]. Security against impersonation under active (and concurrent) attack, however, remained an open question for a while. Indeed, it does not seem possible to prove this under the dl assumption. Eventually, however, security against impersonation under active and concurrent attack was proved by [BP02] under the one more dl (omdl) assumption. However, we need special soundness rather than security under impersonation. Also, we need to show that our strong form

<b>Algorithm <math>\mathcal{K}</math></b> $g \xleftarrow{\$} G^*$ $x \xleftarrow{\$} \mathbb{Z}_p$ $X \leftarrow g^x$ $pk \leftarrow (g, X)$ $sk \leftarrow (g, x)$ Return $(pk, sk)$	<b>Prover</b> Input: $sk = (g, x)$ $y \xleftarrow{\$} \mathbb{Z}_p$ $Y \leftarrow g^y$  $z \leftarrow y + cx \pmod p$	$\xrightarrow{Y}$ $\xleftarrow{c}$ $\xrightarrow{z}$	<b>Verifier</b> Input: $pk = (g, X)$  $c \xleftarrow{\$} \mathbb{Z}_p$  If $g^z = YX^c$ then DEC $\leftarrow$ 1 else DEC $\leftarrow$ 0 Return DEC
---	--	--	--

<b>Alg. pkg</b> $K \xleftarrow{\$} \{0, 1\}^k$ $g \xleftarrow{\$} G^*$ $x \xleftarrow{\$} \mathbb{Z}_p$ $X \leftarrow g^x$ $ppk \leftarrow (K, g, X)$ $psk \leftarrow (K, g, x)$ Return $(ppk, psk)$	<b>Alg. skg(<math>ppk, psk</math>)</b> $(K, g, X) \leftarrow ppk$ $(K, g, x) \leftarrow psk$ $y \xleftarrow{\$} \mathbb{Z}_p$ $Y \leftarrow g^y$ $spk \leftarrow Y$ $ssk \leftarrow Y  y$ Return $(spk, ssk)$	<b>Alg. sgn(<math>psk, ssk, m</math>)</b> $(K, g, x) \leftarrow psk$ $Y  y \leftarrow ssk$ $c \leftarrow H(K, Y  m)$ $z \leftarrow y + cx \pmod p$ $s \leftarrow z$ Return $s$	<b>Alg. vf(<math>ppk, spk, m, s</math>)</b> $(K, g, X) \leftarrow ppk$ $Y \leftarrow spk$ $c \leftarrow H(K, Y  m)$ $z \leftarrow s$ If $g^z = YX^c$ then Return 1 Else Return 0
---	--	--	---

Figure 5: **Schnorr Protocol**. Above,  $G$  is a group of prime order  $p$ , and  $\text{ChSet} = \mathbb{Z}_p$ . The top figure shows the Schnorr identification protocol, while the bottom figure shows the Schnorr-based two-tier scheme obtained by applying our transform to the Schnorr ID protocol.

of special soundness holds, namely that the adversary not only cannot find a pair of accepting transcripts  $(\text{CM}, \text{CH}_1, \text{RP}_1), (\text{CM}, \text{CH}_2, \text{RP}_2)$  with  $\text{CH}_1 \neq \text{CH}_2$  but cannot even find such transcripts with  $\text{CH}_1 = \text{CH}_2$  as long as  $\text{RP}_1 \neq \text{RP}_2$ . We revisit the proof to establish these things. We make use of the fact that the protocol has a “unique answer” property.

**Proposition 5.2** Let  $\text{ID} = (\mathcal{K}, P, \text{ChSet}, V)$  be the Schnorr identification protocol described in Figure 5. Let  $I$  be an adversary against the special soundness of  $\text{ID}$  under concurrent attack. Then there exists an omdl adversary  $A$  such that

$$\text{Adv}_{\text{ID}}^{\text{ss-ca}}(I) \leq \text{Adv}_G^{\text{omdl}}(A).$$

Furthermore the running time of  $A$  is that of  $I$  plus some overhead to compute an inverse and product modulo  $p$ , and if  $I$  activates  $q$  clones, then  $A$  makes  $q + 1$  challenge queries. ■

We remark that the reduction is tight. In contrast, in the reductions showing security against impersonation [BP02],  $\text{Adv}_G^{\text{omdl}}(A)$  is proportional to the square of the probability that  $I$  succeeds in impersonation. This is an advantage to working directly with special soundness rather than with impersonation. We now sketch a proof based on the ideas of [BP02].

**Proof of Proposition 5.2:** The omdl adversary  $A$  gets input  $g \in G^*$ , makes an initial request to the challenge oracle to receive a reply  $W_0$ , and sets  $pk \leftarrow (g, W_0)$ . It then initializes a counter  $a$  to 0 and runs  $I$  on input  $pk$ . When  $I$  asks to activate a new clone,  $A$  increments  $a$ , queries its challenge oracle to receive  $W_a$  and returns this response to  $I$ . When  $I$  sends a challenge  $c_i$  to an activated clone  $P_i$ , adversary  $A$  queries  $z_i \leftarrow \text{DLog}_g(W_i W_0^{c_i})$  and returns  $z_i$  to  $I$ . This provides a perfect simulation for  $I$ , and  $A$  makes strictly fewer discrete log queries than challenge queries. Eventually  $I$  will halt with output a pair  $(Y, c_1, z_1), (Y, c_2, z_2)$  of accepting transcripts. If this output is accepted in  $I$ 's game, then it must be that  $(c_1, z_1) \neq (c_2, z_2)$ .

We now observe that the Schnorr protocol has a special property not needed in [BP02], namely, the unique answer property: for any fixed public key  $X$ , commitment  $Y$ , and challenge  $c$ , there is exactly one response

<p><b>Algorithm <math>\mathcal{K}</math></b>  <math>g_1, g_2 \xleftarrow{\\$} G^*</math>  <math>s_1, s_2 \xleftarrow{\\$} \mathbb{Z}_p</math>  <math>v \leftarrow g_1^{-s_1} g_2^{-s_2}</math>  <math>pk \leftarrow (g_1, g_2, v)</math>  <math>sk \leftarrow (g_1, g_2, s_1, s_2)</math>  Return <math>(pk, sk)</math></p>	<p><b>Prover</b>  Input: <math>sk = (g_1, g_2, s_1, s_2)</math>  <math>r_1, r_2 \xleftarrow{\\$} \mathbb{Z}_p</math>  <math>R \leftarrow g_1^{r_1} g_2^{r_2}</math></p> <p style="text-align: center;"> <math>\xrightarrow{R}</math>  <math>\xleftarrow{e}</math> </p> <p><math>y_1 \leftarrow r_1 + es_1 \pmod p</math>  <math>y_2 \leftarrow r_2 + es_2 \pmod p</math></p> <p style="text-align: center;"> <math>\xrightarrow{(y_1, y_2)}</math> </p>	<p><b>Verifier</b>  Input: <math>pk = (g_1, g_2, v)</math></p> <p style="text-align: center;"> <math>e \xleftarrow{\\$} \mathbb{Z}_p</math> </p> <p>If <math>R = g_1^{y_1} g_2^{y_2} v^e</math>  then <math>DEC \leftarrow 1</math> else <math>DEC \leftarrow 0</math>  Return <math>DEC</math></p>
---	---	---

<p><b>Alg. pkg</b>  <math>K \xleftarrow{\\$} \{0, 1\}^k</math>  <math>g_1, g_2 \xleftarrow{\\$} G^*</math>  <math>s_1, s_2 \xleftarrow{\\$} \mathbb{Z}_p</math>  <math>v \leftarrow g_1^{-s_1} g_2^{-s_2}</math>  <math>ppk \leftarrow (K, g_1, g_2, v)</math>  <math>psk \leftarrow (K, g_1, g_2, s_1, s_2)</math>  Return <math>(pk, sk)</math></p>	<p><b>Alg. skg(ppk, psk)</b>  <math>(K, g_1, g_2, v) \leftarrow ppk</math>  <math>(K, g_1, g_2, s_1, s_2) \leftarrow psk</math>  <math>r_1, r_2 \xleftarrow{\\$} \mathbb{Z}_p</math>  <math>R \leftarrow g_1^{r_1} g_2^{r_2}</math>  <math>spk \leftarrow R</math>  <math>ssk \leftarrow R \  r_1 \  r_2</math>  Return <math>(spk, ssk)</math></p>	<p><b>Alg. sgn(psk, ssk, m)</b>  <math>(K, g_1, g_2, s_1, s_2) \leftarrow psk</math>  <math>R \  r_1 \  r_2 \leftarrow ssk</math>  <math>e \leftarrow H(K, R \  m)</math>  <math>y_1 \leftarrow r_1 + es_1 \pmod p</math>  <math>y_2 \leftarrow r_2 + es_2 \pmod p</math>  <math>s \leftarrow y_1 \  y_2</math>  Return <math>s</math></p>	<p><b>Alg. vf(ppk, spk, m, s)</b>  <math>(K, g_1, g_2, v) \leftarrow ppk</math>  <math>R \leftarrow spk</math>  <math>e \leftarrow H(K, R \  m)</math>  <math>y_1 \  y_2 \leftarrow s</math>  If <math>R = g_1^{y_1} g_2^{y_2} v^e</math> then  Return 1  Else Return 0</p>
---	---	--	---

Figure 6: **Okamoto Protocol**. Above,  $G$  is a group of prime order  $p$ , and  $\text{ChSet} = \mathbb{Z}_p$ . The top figure shows the Okamoto identification protocol, while the bottom figure shows the Okamoto-based two-tier scheme obtained by applying our transform to the Okamoto ID protocol.

$z \in \mathbb{Z}_p$  which is accepted by the verifier, namely for which  $g^z = YX^c$ . (Specifically,  $z = \text{DLog}_g(YX^c)$ .) Since the two transcripts are accepting we have  $g^{z_1} = YW_0^{c_1}$  and  $g^{z_2} = YW_0^{c_2}$ . It follows from the unique answer property that  $c_1 \neq c_2$ , for otherwise the pair of transcripts would be identical.

Using the fact that  $c_1 \neq c_2$ ,  $A$  may recover  $w_0 \leftarrow (z_1 - z_2)(c_1 - c_2)^{-1} \pmod p$  and compute  $w_i \leftarrow z_i - c_i w_0 \pmod p$  for  $1 \leq i \leq q$ . If  $c_i, z_i$  weren't previously defined, we may generate our own  $c_i$  and then compute  $z_i \leftarrow \text{DLog}_g(W_i W_0^{c_i})$  as before, while still making fewer discrete log queries than challenge queries.) It is easy to verify that if  $I$ 's output is accepted in its game, then  $g^{w_i} = W_i$  for  $0 \leq i \leq q$ , and thus  $A$  will win the omdl game whenever  $I$  wins the ss-ca game.

The unique answer property was required to ensure that  $c_1 \neq c_2$  in our proof, as  $A$  has no control of  $I$ 's response. In contrast, the omdl adversary in [BP02] selected its own  $c_1$  and  $c_2$  values uniformly at random to ensure  $c_1 \neq c_2$  with high probability. ■

The two-tier scheme resulting from our FS-based transform instantiated with the Schnorr protocol is very efficient (shown in Figure 5). Generating a secondary key pair takes just one group exponentiation, while signing only requires a multiplication modulo  $p$ . In the context of our uf-cma to suf-cma transform of Section 4, this means that the computational overhead for signing (added cost of signing in the suf-cma scheme compared to that in the uf-cma scheme) is just one group exponentiation and the bandwidth overhead (added length of a signature in the suf-cma scheme compared to that in the uf-cma scheme) is one group element and one integer modulo  $p$ .

**OKAMOTO IDENTIFICATION PROTOCOL.** Okamoto's protocol [Oka92] is illustrated in Figure 6. Its advantage over the Schnorr protocol is that security can be proved under the standard dl assumption rather than the omdl assumption. (Yet in fact the efficiency is not much different from that of the Schnorr protocol as we will see below.) The idea is that there are many secret keys corresponding to a single public key and

witness-indistinguishability [FS90] can be used in the simulation. The protocol was proved in [Oka92] to be secure against impersonation under active attack assuming hardness of the dl problem, and the proof extends to concurrent attacks. However, again, we need special soundness rather than security under impersonation, and in our new, strong form. The Okamoto protocol, however, does not have the unique answer property. But we can still prove the security we need. We now state the result.

**Proposition 5.3** Let  $\text{ID} = (\mathcal{K}, P, \text{ChSet}, V)$  be the Okamoto identification protocol described in Figure 6. Let  $I$  be an adversary against the special soundness of  $\text{ID}$  under concurrent attack. Then there exists a dl adversary  $A$  such that

$$\text{Adv}_{\text{ID}}^{\text{ss-ca}}(I) \leq \frac{1}{p} + \text{Adv}_G^{\text{dl}}(A).$$

Furthermore the running time of  $A$  is that of  $I$  plus the time to compute three inverses and three products modulo  $p$ . ■

Again, the reduction is essentially tight due to working with special soundness, whereas the reduction of [Oka92] to establish security against impersonation incurs the square loss we discussed in the context of Schnorr.

**Proof of Proposition 5.3:** Adversary  $A$  is given input a generator we denote  $g_1$  and a group element  $g_2 = g_1^\alpha$ . Its goal is to find  $\alpha$ . If  $g_2 = 1$  then it outputs 0 and halts, having found  $\alpha$ . Otherwise,  $g_1$  and  $g_2$  are generators. Now it chooses its own  $s_1, s_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and sets  $v \leftarrow g_1^{-s_1} g_2^{-s_2}$ . Then  $pk \leftarrow (g_1, g_2, v)$  is given to  $I$ . But secret key  $sk \leftarrow (g_1, g_2, s_1, s_2)$  is known to  $A$ . So adversary  $A$  can respond to  $I$ 's oracle queries in a straightforward manner, providing a perfect simulation. Eventually  $I$  halts with output  $(R, e, (y_1, y_2)), (R, e', (y'_1, y'_2))$ . As before, we consider the case when  $I$  wins.

In proving the Schnorr protocol, we made use of the fact that it had the unique answer property. However, the Okamoto protocol does not have this property, as there are  $p$  possible valid responses for each fixed  $R$  and  $e$ . Thus we cannot assume that  $e \neq e'$ . Instead, we consider two separate cases.

If  $e \neq e'$ , then  $A$  may solve for  $\alpha$  as described in [Oka92] with high probability. To determine  $\alpha$ ,  $A$  first calculates  $s_1^* = (y_1 - y'_1)(e - e')^{-1} \bmod p$  and  $s_2^* = (y_2 - y'_2)(e - e')^{-1} \bmod p$ , which can be done since  $e \neq e'$ . It then calculates  $\beta = (s_1^* - s_1)(s_2 - s_2^*)^{-1} \bmod p$ , which can be done whenever  $s_2 \neq s_2^*$ . The witness-indistinguishability of the protocol says that it is impossible for  $I$  to have distinguished which  $(s_1, s_2)$  was chosen by  $A$  during its interaction. Then since  $s_2$  is chosen uniformly random, the probability that  $s_2 = s_2^* = (y_2 - y'_2)(e - e')^{-1} \bmod p$  is  $1/p$ . This yields the additive term in our bound. To show that  $g_1^\beta = g_2$ , we first observe that  $v = g_1^{-s_1} g_2^{-s_2} = g_1^{-s_1^*} g_2^{-s_2^*}$ , which is easy to verify with simple algebraic manipulation. We may rearrange the equality to yield  $g_1^{s_1^* - s_1} = g_2^{s_2 - s_2^*}$  and thus  $g_2 = g_1^{(s_1^* - s_1)(s_2 - s_2^*)^{-1}} = g_1^\beta$ . Then  $A$  may output  $\beta$  as its answer.

If, on the other hand,  $e = e'$ , we have  $g_1^{y_1} g_2^{y_2} = g_1^{y'_1} g_2^{y'_2}$ . Assuming  $I$  wins, we know that  $(e, (y_1, y_2)) \neq (e', (y'_1, y'_2))$ . But since  $e = e'$  it must be that  $(y_1, y_2) \neq (y'_1, y'_2)$ . But then since  $g_1^{y_1} g_2^{y_2} = g_1^{y'_1} g_2^{y'_2}$  and  $g_1, g_2$  are generators, it must be that  $y_1 \neq y'_1$  and also  $y_2 \neq y'_2$ . Now  $A$  computes  $\beta = (y_1 - y'_1)(y'_2 - y_2)^{-1} \bmod p$ . Then we have  $g_1^\beta = g_2$  and thus  $A$  can output  $\beta$ . ■

In the two-tier scheme resulting from our FS-transform instantiated with the Okamoto protocol (shown in Figure 6), generating a secondary key pair takes about one group exponentiation (it is a 2-fold multi-exponentiation, which has about the same cost as a single one [MvV97]). Signing requires a couple of multiplications modulo  $p$ . So the computational cost is the same as for Schnorr although security relies only on dl rather than omdl. In the context of our uf-cma to suf-cma transform of Section 4, this means that the computational overhead for signing is again just one group exponentiation. But the bandwidth overhead is one group element and two integers modulo  $p$ , slightly more than when we used the Schnorr scheme.

**ADDITIONAL PROTOCOLS.** Above we have discussed two protocols that meet our ss-ca security requirement. We have however identified several more with the property in question. We omit proofs since they are similar to the ones given here, and instead provide a brief discussion. Figure 8 gives a summary of these protocols and the efficiency of the corresponding two-tier scheme for each. The Fiat-Shamir protocol [FS87], shown in

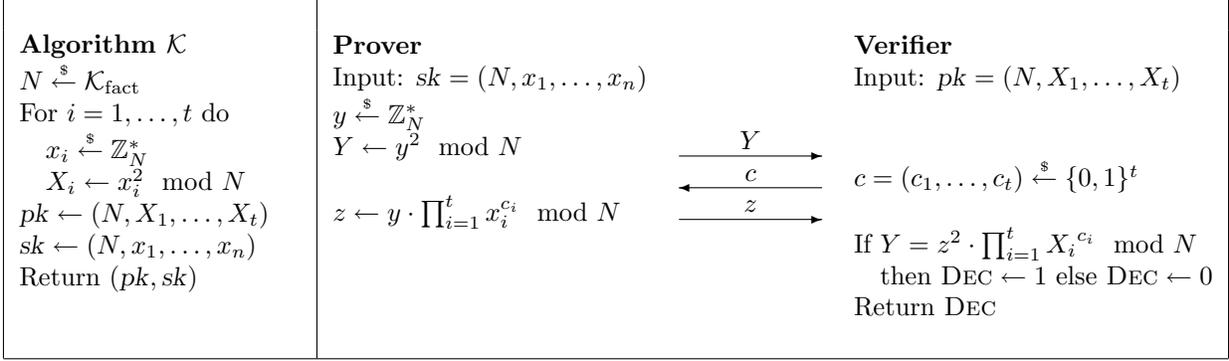


Figure 7: **Fiat-Shamir Protocol**. Above,  $N$  is the product of two distinct primes, generated by some randomized algorithm  $\mathcal{K}_{\text{fact}}$ .

Figure 7, does not meet our strong form of the special soundness requirement. The protocol can be attacked given one honest transcript  $(Y, c, z)$  by outputting the transcripts  $(Y, c, z), (Y, c, -z \bmod N)$ .

The GQ protocol [GQ90] was proved secure against impersonation under concurrent attack in [BP02] under the assumption that RSA is secure against one more RSA inversion (omri). (This is an RSA analogue of the omdl assumption. Both assumptions were introduced in [BNPS03]. Security of GQ under active or concurrent attack under the one-wayness of RSA remains open.) We can extend this proof to show it is ss-ca under the same assumption in the same way that we extended the proof of the Schnorr scheme. This protocol has efficiency similar to Fiat-Shamir yet has small key sizes.

Shamir presented an identity-based identification scheme in [Sha85]. A corresponding standard (i.e. not identity-based) version was presented in [BNN04], along with a variant they called  $\text{Sh}^*$  and proved secure against impersonation under concurrent attack assuming security of RSA under one more inversion. This too can be proved ss-ca secure under the same assumption. The protocol is however a mirror image of GQ and has the same efficiency attributes as the latter.

Then there are pairings-based schemes. Both Hs-SI [BNN04] and ChCh-SI [BNN04] are ss-ca secure under the one more computational Diffie-Hellman assumption (om-cdh). These identification schemes were presented in [BNN04] and are based upon existing IBS (identity-based signature) schemes, namely those of Hess [Hes03] and Cha and Cheon [CC03]. Again, the proof of ss-ca security extends the proofs of security against impersonation of [BNN04].

## 6 Relations between two-tier and one-time schemes

In this section, we explore the relations between two-tier schemes and one-time schemes and discuss the implications of these relations. In particular, we show that we can convert any two-tier scheme into a one-time scheme, and, similarly, we can convert any one-time scheme into a two-tier scheme. We will compare our Fiat-Shamir derived one-time schemes to existing schemes to show the advantages of our newly derived schemes.

**ONE-TIME SCHEMES FROM TWO-TIER SCHEMES.** A two-tier signature scheme yields a one-time signature scheme as a special case (by restricting to a single secondary key). We are interested in this transform because we then obtain FS-based strongly unforgeable one-time signature schemes (using our FS-based constructions of two-tier schemes from Section 5), which have some advantages over existing one-time schemes. We now formally describe how to convert any two-tier signature scheme  $ds = (\text{pkg}, \text{skg}, \text{sgn}, \text{vf})$  into a one-time signature scheme  $DS = (\text{KG}, \text{SGN}, \text{VF})$ . The KG algorithm runs  $\text{pkg}$  to get  $(ppk, psk)$ , runs  $\text{skg}(ppk, psk)$  to get  $(spk, ssk)$ , and returns  $ppk\|spk$  as the public key and  $psk\|ssk$  as the secret key. The remaining algorithms simply use the respective algorithms from the two-tier signature scheme. That is, sign algorithm  $\text{SGN}(psk\|ssk, M)$  returns  $S \xleftarrow{\$} \text{sgn}(psk, ssk, M)$ , and verification algorithm  $\text{VF}(ppk\|spk, M, S)$  returns  $\text{vf}(ppk, spk, M, S)$ . The following theorem says that if  $ds$  is strongly unforgeable then so is  $DS$ .

Base Protocol	Assumption	Secondary Key Generation Cost	Signing Cost	Verification Cost	Size of Secondary Public Key	Signature Size
Schnorr [Sch91]	omdl	1 exp.	1 mult., 1 add.	1 exp.	1 group elt.	1 elt. $\in \mathbb{Z}_p$
Okamoto [Oka92]	dl	2 exp.	2 mult., 2 add.	1 exp.	1 group elt.	2 elt. $\in \mathbb{Z}_p$
GQ [GQ90]	omri	1 exp.	1 exp., 1 mult.	1 exp.	1 group elt.	1 elt. $\in \mathbb{Z}_N$
Sh*-SI [BNN04]	omri	1 exp.	1 exp., 1 mult.	1 exp.	1 group elt.	1 elt. $\in \mathbb{Z}_N$
Hs-SI [BNN04]	om-cdh	1 pairing; 1 exp.	2 mult., 2 add.	2 pairings, 1 exp.	1 group elt.	1 group elt.
ChCh-SI [BNN04]	om-cdh	1 mult.	1 add., 1 mult.	2 pairings	1 group elt.	1 group elt.

Figure 8: **Summary of FS-based two-tier schemes.** Column 1 is the name of the base identification protocol used to construct the two-tier scheme produced by applying the FS transform. Column 2 is the assumption under which the protocol is ss-ca secure. Column 3 gives the cost of generating secondary keys in the two-tier scheme, column 4 gives the cost of signing, and column 5 gives the verification cost. Columns 6 and 7 give the size of the secondary public key and signature, respectively. All schemes have an additional cost of one hash operation for signing and verification.

**Theorem 6.1** Let  $DS = (KG, SGN, VF)$  be the one-time signature scheme associated to two-tier signature scheme  $ds$  as described above. Let  $F$  be an adversary attacking the strong unforgeability of  $DS$  which makes exactly one sign query. Then there exists an adversary  $f$  attacking the strong unforgeability of  $ds$  and making one secondary key query and one sign query, such that

$$\text{Adv}_{DS}^{\text{suf-cma}}(F) \leq \text{Adv}_{ds}^{\text{suf-cma}}(f).$$

Furthermore the running time of  $f$  is the time to run  $F$ . ■

**Proof of Theorem 6.1:** Adversary  $f$  receives as input  $ppk$  and makes an SPKO query to receive  $spk_1$ . It provides the public key  $PK \leftarrow ppk || spk_1$  to  $F$ . When  $F$  makes its sign query on message  $M'$ , adversary  $f$  returns  $S' \stackrel{\$}{\leftarrow} \text{SIGN}_O(1, M')$ . Finally, when  $F$  halts with forgery  $(M, S)$ , adversary  $f$  returns  $(1, M, S)$  as its forgery. This is a valid forgery for  $f$  whenever  $(M, S) \neq (M', S')$  and  $\text{vf}(ppk, spk_1, M, S) = 1$ , which is true whenever  $F$  strongly forges. ■

As an example, applying the above transform to the Schnorr-based two-tier scheme of Figure 5 yields the one-time scheme shown in Figure 11. Note that we have made an optimization in the figure by removing  $g$  from the secret key  $SK$ , as it turns out to be unnecessary in signing a message. To highlight the advantages of this scheme versus existing ones, we revisit the iterated hash one-time scheme of [EGM96], which is the most efficient one-time scheme known.

ITERATED HASH ONE-TIME SIGNATURE SCHEME. We now describe the iterated hash one-time signature scheme of [EGM96]. We will use a family of functions, namely  $H: \{0, 1\}^k \times \{0, 1\}^d \rightarrow \{0, 1\}^d$ . Let  $H^i(K, x)$  mean iterating the hash function  $H(K, \cdot)$  on  $x$  for  $i$  total iterations, or in other words, let  $H^0(K, x) = x$  and  $H^i(K, x) = H(K, H^{i-1}(K, x))$  for  $i \geq 1$ . Associated to the scheme are parameters  $L, t, n$ , and  $T$ , where  $L$  is the message length (e.g.  $L = 160$ ),  $t$  is an integer dividing  $L$ ,  $n = L/t$ , and  $T = 2^t$ . A message is a sequence  $m_1, \dots, m_n$  where each  $m_i$  is an integer in the range  $\{0, \dots, T - 1\}$ . Figure 9 shows the algorithms of the one-time signature scheme  $DS = (KG, SGN, VF)$  associated to these parameters.

This one-time scheme was shown by [EGM96] to be uf-cma secure under the assumption that  $H$  is quasi one-way (briefly, given an image  $y$  for a hash function  $H$ , an adversary attacking quasi one-wayness must find an  $x'$  and  $r$  such that  $H^{r+1}(K, x') = H^r(K, y)$ ). We observe that it is suf-cma secure under the additional assumption that the underlying hash function is collision resistant.

<p><b>Algorithm KG</b></p> $K \xleftarrow{\$} \{0, 1\}^k$ $x_0 \xleftarrow{\$} \{0, 1\}^d$ $y_0 \leftarrow H^{n(T-1)}(K, x_0)$ <p>For <math>i = 1, \dots, n</math> do</p> $x_i \xleftarrow{\$} \{0, 1\}^d$ $y_i \leftarrow H^{T-1}(K, x_i)$ $SK \leftarrow K \  x_0 \  \dots \  x_n$ $PK \leftarrow K \  y_0 \  \dots \  y_n$ <p>Return <math>(PK, SK)</math></p>	<p><b>Algorithm SGN</b>(<math>SK, M</math>)</p> <p>Parse <math>M</math> as <math>m_1 \  \dots \  m_n</math></p> <p>Parse <math>SK</math> as <math>K \  x_0 \  \dots \  x_n</math></p> $s_0 \leftarrow H^{m_1 + \dots + m_n}(K, x_0)$ <p>For <math>i = 1, \dots, n</math> do</p> $s_i \leftarrow H^{T-1-m_i}(K, x_i)$ $S \leftarrow s_0 \  \dots \  s_n$ <p>Return <math>S</math></p>	<p><b>Algorithm VF</b>(<math>PK, M, S</math>)</p> <p>Parse <math>M</math> as <math>m_1 \  \dots \  m_n</math></p> <p>Parse <math>S</math> as <math>s_0 \  \dots \  s_n</math></p> <p>Parse <math>PK</math> as <math>K \  y_0 \  \dots \  y_n</math></p> $b \leftarrow 1$ <p>If <math>H^{n(T-1)-m_1-\dots-m_n}(K, s_0) \neq y_0</math></p> <p style="padding-left: 2em;">then <math>b \leftarrow 0</math></p> <p>For <math>i = 1, \dots, n</math> do</p> <p style="padding-left: 2em;">If <math>H^{m_i}(K, s_i) \neq y_i</math></p> <p style="padding-left: 4em;">then <math>b \leftarrow 0</math></p> <p>Return <math>b</math></p>
---	--	--

Figure 9: **Iterated hash one-time signatures scheme.** Associated to the scheme are parameters  $L$ ,  $t$ ,  $n$ , and  $T$ , where  $L$  is the message length (e.g.  $L = 160$ ),  $t$  is an integer dividing  $L$ ,  $n = L/t$ , and  $T = 2^t$ . A message is a sequence  $m_1, \dots, m_n$  where each  $m_i$  is an integer in the range  $\{0, \dots, T - 1\}$ .

COMPARISON TO SCHNORR-BASED ONE-TIME SCHEME. The point of presenting this scheme is to compare to our own Fiat-Shamir derived schemes (specifically, using Schnorr as the base identification protocol). The iterated hash one-time signature scheme has several adjustable parameters, namely the length of the message to be signed and the number of blocks within each message (each block is an input to the hash function). There is a tradeoff between key and signature sizes versus computational cost depending upon the choice of these parameters. (For a fixed message size, a smaller block size will mean larger key and signature sizes but cheaper computational cost.) Thus, to compare this scheme to our two-tier based Schnorr one-time scheme, we will consider an instance of the scheme with  $d = L = 160$  (i.e. the hash function takes 160-bit input and produces 160-bit output, and the message input length is 160 bits). In practice, we will use a hash function such as SHA-1 in place of  $H$  so that there is no key  $K$ . Figure 10 shows the public key, private key, and signature bit lengths, along with the cost of signing, verification, and key generation for all possible values of  $t$  with a fixed  $d = L = 160$ .

In comparison to the iterated-hash one-time scheme, consider the Schnorr-based one-time scheme constructed by applying the FS-based transform to create a two-tier signature scheme and then applying the above transform to create the one-time scheme. The resulting scheme is shown in Figure 11. Let  $G$  be a 160-bit elliptic curve group, with  $|p| \approx 160$ . Key generation in our Schnorr-based one-time scheme requires two group exponentiations, or about 6600 hash computations (recall that a group exponentiation with the crypto++ library is about 3300 hashes for an unoptimized implementation [Dai]). Signing requires only one hash computation and an operation modulo  $p$ . Verification requires one group exponentiation and one group multiplication. The public and secret key lengths for these parameters are both  $160 \cdot 3 = 480$  bits, and signature size is 160 bits.

From Figure 10 one may see that it is unreasonable to achieve similar lengths in the iterated hash scheme, as the cost of computation is infeasible for such short keys. Alternatively, choosing  $t = 10$  so that key generation costs at least as much in this scheme as in the Schnorr scheme yields 2720-bit keys and signatures. This is significantly larger than the 480 bits required by Schnorr. Furthermore, at this point the cost of signing is also significantly cheaper for Schnorr (16368 hashes in the iterated hash scheme versus one hash and one operation modulo  $p$  in the Schnorr scheme). Thus, our Schnorr one-time scheme has efficiency gains in both space (in the size of the keys and signature) and time (in signing) over the conventional scheme of [EGM96]. Note that we would get the same efficiency gains using the standard Schnorr signature scheme instead of our scheme, but the proof of the former uses random oracles.

Recall that our Schnorr one-time scheme relies only upon the difficulty of the one-more-discrete-log (omdl) problem and the collision resistance of the underlying hash function. We also observe that we would obtain similar efficiency gains (with slightly longer key and signature lengths) over the scheme of [EGM96] if we were to compare to our FS-derived Okamoto-based one-time signature scheme, and the security of this scheme relies only upon the hardness of the discrete log problem and the collision resistance of the underlying hash

$t$	Public Key, Private Key, and Signature bit-length $((160/t + 1) \cdot 160)$	Number of Hash Computations for Signing, Verification, and Key Generation $(\approx 160/t \cdot (2^t - 1))$
1	25760	160
2	12960	240
4	6560	600
5	5280	992
8	3360	5100
10	2720	16368
16	1760	655,350
20	1440	8,388,600
32	960	$8 \times (2^{32} - 1)$
40	800	$4 \times (2^{40} - 1)$
80	480	$2 \times (2^{80} - 1)$
160	320	$2^{160} - 1$

Figure 10: **Efficiency of Iterated Hash Scheme.** We fix the following parameters of the iterated hash one-time signature scheme:  $d = 160$  and  $L = 160$ . This chart considers all possible values for  $t$  and displays the corresponding public and private key bit-lengths, signature bit-length, and the number of hash computations required for signing and verification.

Algorithm KG	Algorithm SGN( $SK, M$ )	Algorithm VF( $PK, M, S$ )
$K \xleftarrow{\$} \{0, 1\}^k; g \leftarrow G^*$	Parse $SK$ as $(K, x, y, Y)$	Parse $PK$ as $(K, g, X, Y)$
$x \xleftarrow{\$} \mathbb{Z}_p; X \leftarrow g^x$	$c \leftarrow H(K, Y \  M)$	$c \leftarrow H(K, Y \  M)$
$y \xleftarrow{\$} \mathbb{Z}_p; Y \leftarrow g^y$	$z \leftarrow y + cx \pmod p$	$z \leftarrow S$
$PK \leftarrow (K, g, X, Y)$	Return $z$	If $g^z = YX^c$ then return 1
$SK \leftarrow (K, x, y, Y)$		Return 0
Return $(PK, SK)$		

Figure 11: **Our Schnorr-based one-time signature scheme.**

function.

TWO-TIER SCHEMES FROM ONE-TIME SCHEMES. A one-time signature scheme yields a two-tier signature scheme by keeping the primary keys empty and using the one-time scheme to generate secondary keys. More formally, we now describe how to convert any one-time signature scheme  $DS = (KG, SGN, VF)$  into a two-tier signature scheme  $ds = (pkg, skg, sgn, vf)$ . The  $pkg$  algorithm returns  $(\varepsilon, \varepsilon)$ . The remaining algorithms simply use the respective algorithms from the one-time signature scheme. That is, the secondary key algorithm returns  $(spk, ssk) \xleftarrow{\$} KG$ , sign algorithm  $sgn(\varepsilon, ssk, m)$  returns  $s \xleftarrow{\$} SGN(ssk, m)$ , and verification algorithm  $vf(\varepsilon, spk, m, s)$  returns  $VF(sp, m, s)$ . The following theorem says that if  $DS$  is strongly unforgeable then so is  $ds$ .

**Theorem 6.2** Let  $ds = (pkg, skg, sgn, vf)$  be the two-tier signature scheme associated to one-time signature scheme  $DS$  as above. Let  $f$  be an adversary attacking the strong unforgeability of  $ds$  and making  $q$  SPKO key queries, where  $q \geq 1$ . Then there exists an adversary  $F$  attacking the strong unforgeability of  $DS$ , such that

$$\text{Adv}_{ds}^{\text{suf-cma}}(f) \leq q \cdot \text{Adv}_{DS}^{\text{suf-cma}}(F).$$

Furthermore the running time of  $F$  is the time to run  $f$  plus a constant amount of overhead.  $\blacksquare$

**Proof of Theorem 6.2:** Adversary  $F$  receives as input  $PK$  and chooses a random key query  $q'$  from  $\{1, \dots, q\}$ . It provides to  $f$  the primary public key  $\varepsilon$ . It responds to  $f$ 's SPKO queries by running KG for each request and returning the corresponding public key, except on the  $q'$ -th query, on which it returns  $PK$ . It responds to  $f$ 's sign queries according to the algorithm, using the secret keys it has generated, except on a query for key  $q'$ . Here it uses its own signing oracle to produce the signature  $s$ . Since  $f$  may make a sign query for each key at most once, then  $F$  also makes at most one query to its sign oracle. We denote by  $m_i$  the  $i$ -th message queried by  $f$  and  $s_i$  the corresponding signatures provided during the simulation.

When  $f$  halts with forgery  $(l, m, s)$ ,  $F$  checks that  $l = q'$  and aborts if this condition is false. Otherwise,  $F$  outputs  $(m, s)$ . If  $f$ 's forgery is valid, then  $\text{vf}(\varepsilon, PK, m, s) = 1$  and  $(m, s) \neq (m_i, s_i)$ . It follows that  $\text{VF}(PK, m, s) = 1$  and thus  $(m, s)$  is a forgery for  $F$ .

Since the choice of  $q'$  has no effect on  $f$ 's behavior, it follows that

$$\text{Adv}_{\text{DS}}^{\text{suf-cma}}(F) \geq \frac{1}{q} \cdot \text{Adv}_{\text{ds}}^{\text{suf-cma}}(f).$$

Furthermore, the running time of  $F$  is that of  $f$  plus a constant amount of overhead. ■

From known constructions of strongly unforgeable one-time schemes, and in particular from the above-mentioned iterated hash one-time scheme, we now obtain new constructions of two-tier schemes. The relative merits compared to our FS-based two-tier schemes are analogous to what was discussed in the one-time case. Namely, for parameters where the computational costs are the same, the FS-based schemes have smaller key sizes. However, if key sizes are not a concern, there are parameters for which the computational costs of these one-time based two-tier schemes are lower than the computational costs of our FS-based two-tier schemes.

## References

- [AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer-Verlag, Berlin, Germany.
- [BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer-Verlag, Berlin, Germany.
- [BM96] Daniel Bleichenbacher and Ueli Maurer. On the efficiency of one-time digital signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 196–209, Kyongju, Korea, November 3–7, 1996. Springer-Verlag, Berlin, Germany.
- [BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS 06*, pages 390–399, Alexandria, Virginia, USA, November 7–11, 2006. ACM Press.
- [BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 268–286, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.
- [BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.
- [BP02] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177, Santa Barbara, CA, USA, August 18–22, 2002. Springer-Verlag, Berlin, Germany.

- [BSW06] Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In Moti Yung, editor, *PKC 2006*, volume 3958 of *LNCS*, pages 229–240, New York, NY, USA, April 24–26, 2006. Springer-Verlag, Berlin, Germany.
- [CC03] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap Diffie-Hellman groups. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 18–30, Miami, USA, January 6–8, 2003. Springer-Verlag, Berlin, Germany.
- [CD95] Ronald Cramer and Ivan Damgård. Secure signature schemes based on interactive protocols. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 297–310, Santa Barbara, CA, USA, August 27–31, 1995. Springer-Verlag, Berlin, Germany.
- [CHK06] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):915–942, 2006.
- [Cra96] Ronald Cramer. Modular design of secure yet practical cryptographic protocols. Ph.D. thesis, CWI and University of Amsterdam, November 1996.
- [Dai] Wei Dai. Crypto ++ library. <http://www.cryptopp.com/>.
- [DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
- [DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In *TCC 2005*, volume 3378 of *LNCS*, pages 188–209, Cambridge, MA, USA, February 10–12, 2005. Springer-Verlag, Berlin, Germany.
- [EGM96] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996.
- [FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.
- [FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer-Verlag, Berlin, Germany.
- [FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426, Seattle, Washington, USA, May 15–17, 1990. ACM Press.
- [GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, February 1996.
- [GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115, Cambridge, Massachusetts, USA, October 11–14, 2003. IEEE Computer Society Press.
- [GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
- [Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
- [GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, Davos, Switzerland, May 25–27, 1988. Springer-Verlag, Berlin, Germany.
- [GQ90] Louis C. Guillou and Jean-Jacques Quisquater. A “paradoxical” identity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 216–231, Santa Barbara, CA, USA, August 21–25, 1990. Springer-Verlag, Berlin, Germany.

- [Hes03] Florian Hess. Efficient identity based signature schemes based on pairings. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 310–324, St. John’s, Newfoundland, Canada, August 15–16, 2003. Springer-Verlag, Berlin, Germany.
- [HWZ06] Qiong Huang, Duncan S. Wong, and Yiming Zhao. Generic transformation to strongly unforgeable signatures. Cryptology ePrint Archive, Report 2006/346, 2006. <http://eprint.iacr.org/2006/346>.
- [Lin06] Yehuda Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. *Journal of Cryptology*, 19(3):359–377, 2006.
- [MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures. In *ACM CCS 01*, pages 245–254, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.
- [MvV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.
- [Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO’92*, volume 740 of *LNCS*, pages 31–53, Santa Barbara, CA, USA, August 16–20, 1992. Springer-Verlag, Berlin, Germany.
- [OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *CRYPTO’98*, volume 1462 of *LNCS*, pages 354–369, Santa Barbara, CA, USA, August 23–27, 1998. Springer-Verlag, Berlin, Germany.
- [PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
- [PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, Chennai, India, December 4–8, 2005. Springer-Verlag, Berlin, Germany.
- [Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
- [Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO’84*, volume 196 of *LNCS*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1985. Springer-Verlag, Berlin, Germany.
- [SPW07] Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In Masayuki Abe, editor, *CT-RSA 2007*, volume 4377 of *LNCS*, pages 357–371, San Francisco, CA, USA, February 5–9, 2007. Springer-Verlag, Berlin, Germany.
- [ST01] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 355–367, Santa Barbara, CA, USA, August 19–23, 2001. Springer-Verlag, Berlin, Germany.
- [TOO06] Isamu Teranishi, Takuro Oyama, and Wakaha Ogata. General conversion for obtaining strongly existentially unforgeable signatures. In *INDOCRYPT 2006*, LNCS, pages 191–205. Springer-Verlag, Berlin, Germany, 2006.
- [Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.