Construction of Universal Designated-Verifier Signatures and Identity-Based Signatures from Standard Signatures[1]

Siamak F Shahandshti † Reihaneh Safavi-Naini ‡

† School of Comp. Sci. and Soft. Eng., University of Wollongong, Australia
http://www.uow.edu.au/∼sfs166

‡ Department of Computer Science, University of Calgary, Canada
http://www.cpsc.ucalgary.ca/∼rei

December 11, 2007

Abstract

We give a generic construction of universal designated-verifier signature schemes from a large class $\mathbb{C}$ of signature schemes. Our constructions are comparable in terms of cost and size to their counterparts, while offering the following two main attractive features: Firstly, our constructions are provably non-delegatable beside being DV-unforgeable and non-transferable. Secondly, in our constructions, the signer and the designated verifier can choose different cryptographic settings of their own independently. We also propose a generic construction of identity-based signature schemes from any signature scheme in $\mathbb{C}$ and prove the construction secure against adaptive chosen message and identity attacks. We discuss possible extensions of our constructions to universal multi-designated-verifier signatures, hierarchical identity-based signatures, identity-based universal designated verifier signatures, and identity-based ring signatures from any signature in $\mathbb{C}$.

Keywords: Designated Verifier Signature, Identity-Based Signature, Digital Signature Schemes, Signature of Knowledge, Generic Construction

1 Introduction

Universal Designated-Verifier Signatures. *Designated verifier proofs* and *designated verifier signatures* (DVS) were proposed by Jakobsson et al. [JSI96] as proofs and signatures that will only convince a specific verifier. The idea is that such proofs/signatures can be constructed by both the prover/signer on one hand and the verifier on the other. When a verifier receives such a proof/signature, since he knows that he has not constructed it himself, he will be convinced. However the verifier cannot convince a third party by showing him what he has received, since it could have been the verifier himself who generated the designated proof/signature.

Universal designated-verifier signatures (UDVS) were first proposed by Steinfeld et al. [SBWP03] with the goal of protecting users' privacy in certification systems. In such a scheme, a user Alice is issued a signed certificate by a certificate issuer. Later on, when Alice wishes to send her certificate to a verifier Bob, she uses Bob's public key to transform the issuer's signature into a *designated signature* for Bob. Bob can verify the issuer's signature by verifying the validity of the designated signature. However, since his secret key allows him to construct valid designated signatures by himself, he is unable to convince a third party that the certificate has been signed by the issuer. The main difference between a DVS and a UDVS scheme is that in the former, only the *signer* has the ability to designate signatures, whereas in the latter, *everyone* can designate signatures.

In their work, Steinfeld et al. also proposed security definitions for UDVS schemes and a concrete scheme based on bilinear group pairs [SBWP03]. In [LWB05] Lipmaa et al. argued that the original security definition in

[1] This is the full version of the article appearing in PKC '08 with the same title.

[SBWP03] was not sufficient to capture the verifier-designation concept and introduced a new security notion, called *non-delegability*. They have shown that in some UDVS schemes including Steinfeld et al's [SBWP03], the issuer can delegate his signing ability - with respect to a fixed designated verifier - to a third party, without revealing his secret key or even enabling the third party to sign with respect to other designated verifiers. They argue that, in many scenarios, such delegation property is undesirable and must be prevented. An example of such scenario is when Alice designates university's signature on her student card to use a certain gym for students. If for some reason, the university's relation with the gym is

Since she uses her card for other services as well, e.g. to borrow a book from the university library, to enter her research lab, and to get a train discount, she is not willing to give her card to her friend Alex. However, if the UDVS in use is delegatable, she will be able to provide Alex with a delegated card that only enables him to use the gym instead of her.

None of the many UDVS schemes proposed to date, except a recent scheme of Huang et al. [HSMW06], has treated non-delegatability as a security requirement for their scheme. Furthermore, the results of Lipmaa et al. [LWB05] and later results of Li et al. [LLP05] show that many of the proposed UDVS schemes are delegatable, including the scheme from [SBWP03] and one of the schemes from [SWP04].

Our Contributions on UDVS. We give a generic construction of secure UDVS scheme from a large class of signature schemes (satisfying certain properties). We use a definition of security that includes the original security notions of Steinfled et al, i.e. *unforgeability* and *non-transferability privacy*, and also the notion of *non-delegatability* inspired by the work of Lipmaa et al. [LWB05] and adapted to UDVS. We define non-delegatability in terms of a 'proof of knowledge' of either a signature on the message or the designated verifier's secret key. This definition guarantees that only Alice or Bob can construct valid designated signatures, and hence they are not able to delegate this ability to others without revealing Alice's certificate or Bob's secret key.

To construct non-delegatable UDVS schemes, we will use Jakobsson et al's original intuition behind the concept of verifier designation [JSI96]: *"Instead of proving Θ, Alice will prove the statement: Either Θ is true, or I am Bob."* In UDVS schemes, what Alice wants to prove to Bob is the validity of her certificate. Thus, a very natural construction of a UDVS would simply be the non-interactive version of a proof of the following statement by Alice: *"Either my certificate is valid, or I am Bob."* Such a signature can be constructed in the following steps: first pick a protocol for proof of knowledge of Alice's certificate and another for proof of knowledge of Bob's secret key, then construct a protocol for proof of knowledge of Alice's certificate *or* Bob's secret key by combining the two protocols via e.g. techniques of Cramer et al. [CDS94], and at last, make the resulting protocol non-interactive via e.g. Fiat-Shamir transform [FS86]. In fact, it is intuitively clear that such a construction yields a secure UDVS scheme, if both of the underlying protocols are honest-verifier zero-knowledge (HVZK) proofs of knowledge. However, efficient protocols for HVZK proof of knowledge of a signature on a message are only known for a small group of signature schemes.

We propose a construction of UDVS schemes that works for any combination of a signature in class $\mathbb{C}$ of signature schemes and all verifiers key pairs that belong to a class $\mathbb{K}$, and prove its security in the above sense, in the Random Oracle Model. The class $\mathbb{C}$ of signatures that can be used in our construction includes signature schemes such as RSA-FDH [BR96], Schnorr [Sch91], modified ElGamal [PS00a], BLS [BLS01], BB [BB04], Cramer-Shoup [CS00], and both schemes proposed by Camenisch and Lysyanskaya [CL02, CL04]. Class $\mathbb{K}$ is the set of all key pairs for which there exist protocols for HVZK proofs of knowledge of the secret key corresponding to a public key and includes public and private key pairs of RSA cryptosystem, GQ identification scheme [GQ88], and discrete-log based public and private key pairs.

Our construction has a number of attractive features. Firstly, it is generic and security proofs guarantee security of a large class of UDVS schemes obtained from standard signature schemes that are members of the class $\mathbb{C}$. Note that the only other known non-delegatable UDVS due to Huang et al. [HSMW06] is in fact an instance of our construction. Secondly, the construction does not limit the signer and the verifier to have 'compatible' settings: the construction works for any choice of signer and verifier settings as long as the signature scheme is a member of class $\mathbb{C}$ and the verifier key belongs to the class $\mathbb{K}$. All previous constructions only work for a specific combination of signature schemes and verifier key pairs.

Identity-Based Signatures. Identity-based cryptography was proposed by Shamir in [Sha84], where he also proposed an *identity-based signature* (IBS) scheme. In an IBS scheme, there is an authority with a key pair: a

master secret key and a master public key, who generates for each user a user secret key based on the user's identity. A user can use its user secret key to sign messages. Signatures can be verified against the identity of the signer and the master public key.

There are two known generic constructions of IBS. The first is due to Bellare et al. [BNN04], which generalizes an earlier construction of Dodis et al. [DKXY03]. Bellare et al. define a class of schemes called *convertible* (in analogy with *trapdoor* schemes defined by Dodis et al.) and show how to construct identity-based signature schemes from signature schemes in this class. They showed that a large number of previously proposed schemes are in fact instances of their generic construction. However, as noted by the authors, there are some IBS schemes, including Okamoto's discrete logarithm based IBS [Oka92] (called OkDL-IBS by Bellare et al.) and a new IBS scheme proposed in [BNN04] (called BNN-IBS), that are not instances of their generic construction.

The other generic construction is the one of Kurosawa and Heng [KH04]. Their construction requires an efficient zero-knowledge protocol for proof of knowledge of a signature, which makes their construction applicable to only a few schemes such as RSA-FDH and BLS.

Our Contributions on IBS. We propose a construction of IBS schemes from any signature in the aforementioned class $\mathbb{C}$ and prove the construction secure against adaptive chosen message and identity attacks. In our construction, a user secret key is basically a signature of the authority on the user's identity and an identity-based signature is generated as follows: the user constructs a proof of knowledge of her secret key (i.e. the authority's signature on her identity) and then transforms it into a signature on a message using the Fiat-Shamir transform. For signature schemes with efficient zero-knowledge protocols for proof of knowledge of a signature, our constructions will become the same as those of Kurosawa and Heng [KH04]. Thus, our constructions can be seen as a generalization of theirs.

Many previous IBS schemes can be seen as instances of our generic construction such as those of Fiat and Shamir from [FS86], Guillou and Quisquater from [GQ88], Shamir from [Sha84], pairing-based schemes from [SOK00, Hes02, CC03, Yi03, BLMQ05, HCW05] and basically all the convertible IBS schemes constructed in [BNN04]. Both OkDL-IBS and BNN-IBS, which are not captured by generic constructions of Bellare et al, fit as instances of our generic construction as well. However, all the IBS schemes that we construct are proved secure in the *Random Oracle model (ROM)* [BR93], thus ROM-free constructions such as the folklore *certificate-based* IBS schemes formalized in [BNN04] and the scheme of Paterson and Schuldt [PS06] are not captured by our framework.

Further Contributions. Our constructions of UDVS schemes can be naturally extended to (non-delegatable) *universal multi-designated-verifier signatures*. Furthermore, we observe that our identity-based constructions support a *nesting-like* property in the sense that a user can act as a new key generation authorithy and issue keys for other users. This fact enables extensions of our IBS constructions to *hierarchical identity-based signatures* out of any signature scheme in the class $\mathbb{C}$. We will also point out the possibility of generic construction of (non-delegatable) *identity-based universal designated verifier signatures* and *identity-based ring signatures* from any signature in $\mathbb{C}$ using our techniques.

1.1 Related Work

As mentioned before, UDVS schemes were first proposed by Steinfeld et al. in [SBWP03]. In the same work, they also proposed security definitions and a concrete scheme based on bilinear group pairs. In a later work, they proposed extensions of Schnorr and RSA signatures into UDVS schemes [SWP04]. Subsequently, other pairing-based schemes were proposed in [ZFI05] and [Ver06], and Laguillaumie et al. introduced 'Random Oracle free' constructions [LLQ06]. Besides, many other UDVS schemes with various flavors were constructed (e.g. interactive [BSS05], multi-verifier [NSM05], identity-based [ZSMC05], with aggregation [MT05], ring [LW06], and restricted [HSMZ06]).

Our constructions are very close to Goldwasser and Waisbard's generic constructions of *designated confirmer signatures* in [GW04]. They also use protocols for proof of knowledge of a signature as a tool for their constructions. They also present such protocols for a number of signature schemes including Goldwasser-Micali-Rivest

[GMR88], Gennaro-Halevi-Rabin [GHR99], and Cramer-Shoup [CS00]. This shows that the above signatures are in class $\mathbb{C}$.

A highly related area is that of *ring signatures*. Generic constructions of ring signatures as Fiat-Shamir transformed proofs of knowledge of one-out-of-n secret keys are known for a long time. Our techniques deal with the very similar but different concept of proofs of knowledge of signatures on known messages. Although protocols for proof of knowldge of a secret key corresponding to a public key are more studied and well-known, the counterpart field of proof of knowledge of a signature corresponding to a message and a public key has been less studied independently.

Our ID-based signature is actually an instance of a more general concept, *Signatures of Knowledge*, recently redefined and formalized by Chase and Lysyanskaya [CL06]. A signature of knowledge on a message guarantees that a signer who knows a witness of an NP language has signed the message. Having defined such a signature, conventional signatures will be an instance of such a generic definition, where a signature guarantees that a signer who knows the secret key corresponding to a known public key has signed the message. Our ID-based signature is an instance of a signature of knowledge, in which a signature guarantees that a signer who knows a signature of the key generating authority on his identity, has signed the message.

Our identity-based signature can also be seen as the signature counterpart of *hidden credentials* [HBSO03]. In hidden credentials, Alice encrypts a message in a way that Bob can only decrypt it if he has a certain credential from Chris, i.e. the credential acts as the decryption key. In our identity-based signatures, Bob receives a signature which guarantees that an entity who has a certain credential, i.e. the corresponding user secret key, from Chris, i.e. the key generating authority has signed it.

It is also worth to mention the previous constructions of the identity-based universal deignated verifier signatures by Zhang et al. [ZSMC05] and universal multi-designated-verifier signatures by Ng et al. [NSM05], which are unfortunately both delegatable. Our generic constructions of the above schemes, as mentioned before, guarantee non-delegatability.

2 Preliminaries

2.1 Notation

We use different fonts to denote $\mathsf{Algorithms}$, SECURITY NOTIONS, and $\mathcal{O}racles$, respectively. We also denote the internal state of an algorithm X by St_{X} and the empty string by ε. We will also use a handful of different arrows to denote different things. These are summarized in Table 1.

Table 1: Notation used in the paper

$x \leftarrow a$	a is assigned to x	$x \leftarrow \mathsf{X}(a; \mathcal{O})$	X with input a and access to oracle $\mathcal{O}$ is run and outputs x
$x \overset{N}{\leftarrow} a$	a mod N is assigned to x		
$x \overset{\$}{\leftarrow} X$	x is chosen randomly from X	$\mathsf{A} \overset{a}{\longrightarrow\!\!\!\!\!\rightarrow} \mathsf{B}$	a is sent from A to B

2.2 Proofs of Knowledge

Consider an *NP problem* P. The set of all the pairs consisting of an *instance* Ins of P and its corresponding *solution* Sol, i.e. (Ins, Sol), form a *relation* that we call an *NP relation*. Now, consider an NP relation Rel. Membership of this relation can be decided in polynomial time. Let Rel be the corresponding membership deciding algorithm. Then, a pair (Pub, Sec) belongs to Rel if and only if $\mathsf{Rel}(Pub, Sec)$. Following the works of Camenisch and Stadler [CS97a], we will use the following notation for showing a protocol for *proof of knowledge*

$$\mathsf{PoK}\{Sec : \mathsf{Rel}(Pub, Sec)\} \quad ,$$

where the prover proves knowledge of her secret Sec corresponding to a publicly known Pub, s.t. $(Pub, Sec) \in Rel$. Technically speaking, Sec is the private input to the prover algorithm and Pub is the public input of the protocol. We will follow the convention that all the secret inputs are collectively denoted by Sec and shown before the colon (:) and all the remaining variables, functions, sets, etc. appearing after the colon are assumed to be the public inputs, collectively denoted by Pub.

A *public-coin* protocol is a protocol in which the verifier chooses all its messages during the protocol run randomly from publicly-known sets. A three-move public-coin protocol can be written in the so called *canonical* form as shown in Figure 1. The prover algorithm in this case will consist of a pair of algorithms, respectively for so-called *committing* and *responding*, denoted by $\mathsf{P} = (\mathsf{Cmt}, \mathsf{Rsp})$. The verifier algorithm, in turn, will consist of a pair of algorithms, respectively for so-called *challenging* and *deciding*, denoted by $\mathsf{V} = (\mathsf{Chl}, \mathsf{Dcd})$, where the challenging algorithm is limited to only drawing a challenge randomly from a publicly-known set, called the *challenge space*. As we mentioned before, we will denote by St_{P} the internal state of the prover. The work-flow of the algorithm is as follows. In the first move, the prover runs the algorithm Cmt to compute the *commitment* Cmt and sends it to the verifier. Then the verifier chooses a random *challenge* Chl from a challenge space $ChSp$ and sends it back in the second move. The prover will then compute a *response* Rsp according to the algorithm Rsp based on the information from the first run and the challenge, and then send Rsp to the verifier. Algorithm Dcd will be run by the verifier at the end to compute a *decision* d based on the commitment, the challenge and the response.

$$
\begin{array}{lcl}
(St_{\mathsf{P}}, Cmt) \leftarrow \mathsf{Cmt}\,(Sec, Pub) & \xrightarrow{\;Cmt\;} & \\
 & \xleftarrow{\;Chl\;} & Chl \xleftarrow{\$} ChSp \\
Rsp \leftarrow \mathsf{Rsp}\,(St_{\mathsf{P}}, Chl) & \xrightarrow{\;Rsp\;} & \\
 & & d \leftarrow \mathsf{Dcd}\,(Pub, Cmt, Chl, Rsp)
\end{array}
$$

Figure 1: A canonical three-move public-coin protocol for proof of knowledge

Let's denote the *transcript* of a protocol run in Figure 1 by $Tr = (Cmt, Chl, Rsp)$. The protocol is said to have the *honest-verifier zero-knowledge* property (HVZK from now on) [GMR89], if there exists an algorithm that is able to *simulate* transcripts that are indistinguishable from the ones of the real protocol runs without the knowledge of the secret. The protocol is said to have the *special soundness* property (SpS from now on) as described in [CDS94], if there also exists an algorithm that is able to *extract* the secret from two transcripts of the protocol with the same commitment and different challenges. We denote these as the following, respectively:

$$Tr \leftarrow \mathsf{TrSim}\,(Pub) \quad \text{and} \quad Sec \leftarrow \mathsf{Ext}\,(Pub, Tr, Tr') \ ,$$

where $Tr = (Cmt, Chl, Rsp)$ and $Tr' = (Cmt', Chl', Rsp')$ are such that $Cmt = Cmt'$ but $Chl \neq Chl'$. A three-move public-coin protocol with both the HVZK and SpS properties is usually called a Σ protocol. Examples of Σ protocols for proof of knowledge are the GQ protocol [GQ88] and the Schnorr protocol [Sch91].

2.3 Proofs of Disjunctive Knowledge

Cramer et al. have shown how to extend Σ protocols to *witness indistinguishable (WI from now on)* Σ protocols for proving knowledge of (at least) t out of n values using secret sharing schemes [CDS94]. They call such protocols *proofs of partial knowledge*. Witness indistinguishability guarantees that even a cheating verifier will not be able to tell which t-subset of the n values the prover knows. Thus, the transcripts of different runs of the protocol with different t-subsets as prover input will be indistinguishable from one another.

An instance of such partial proofs of knowledge that we find useful here is a WI proof of knowledge of one out of two, which we call a *proof of disjunctive knowledge*. These proofs were also observed by Camenisch and Stadler [CS97b] for discrete logarithms. In line with the above, we will use the following notation to show such proofs: to show a protocol for proof of knowledge of a value Sec_1 such that $\mathsf{Rel}_1\,(Pub_1, Sec_1)$ or a value Sec_2 such that $\mathsf{Rel}_2\,(Pub_2, Sec_2)$, we use the notation

$$\mathsf{PoK}\,\{(Sec_1 \vee Sec_2) :\ \mathsf{Rel}_1\,(Pub_1, Sec_1)\ ,\ \mathsf{Rel}_2\,(Pub_2, Sec_2)\ \}\,.$$

The Σ protocol for proof of knowledge of Sec_1 or Sec_2 corresponding to $Pub = (Pub_1, Pub_2)$ can be constructed in the canonical form using simple techniques. Both HVZK and SpS properties are also inherited by the constructed proof of disjunctive knowledge. For specifics refer to Appendix F.

2.4 The Fiat-Shamir Transform

Fiat and Shamir proposed a method for transforming (interactive) three-move public-coin protocols into non-interactive schemes [FS86]. The idea is to replace the verifier with a hash function and the rationale behind it is that all the verifier does in such a protocol is providing some sort of unpredictable challenge that can be mimicked by a Random Oracle hash function. This idea can be applied in two different ways, depending on what one includes in the hash function argument. One way is to set the challenge as the hash of the concatenation of the public inputs and the commitment, i.e. $Chl \leftarrow H(Pub \parallel Cmt)$. This way we will get a *non-interactive proof of knowledge*. If such a transform is applied to the protocol in Figure 1 using the Random Oracle hash function $H : \{0,1\}^* \mapsto ChSp$, the resulting non-interactive proof scheme will be as in Figure 2, with the algorithms NIPoK and NIVoK for non-interactive proof and verification of knowledge, respectively. Here, π is a non-interactive proof that can be verified off-line and publicly. HVZK and SpS properties for non-interactive proofs are defined similarly to their counterparts for interactive proofs. Pointcheval and Stern's *Forking Lemma* [PS00a] can be used to easily prove in the Random Oracle Model that the Fiat-Shamir construction has both the HVZK and SpS properties if the original interactive proof has the corresponding properties.

```
Algorithm NIPoK(Pub, Sec)
  (St_P, Cmt) ← Cmt(Sec, Pub)
  Chl ← H(Pub ‖ Cmt)
  Rsp ← Rsp(St_P, Chl)
  π ← (Cmt, Rsp)
  return π
```

```
Algorithm NIVoK(Pub, π)
  Chl ← H(Pub ‖ Cmt)
  d ← Dcd(Cmt, Chl, Rsp)
  return d
```

Figure 2: The non-interactive proof scheme from applying Fiat-Shamir to the protocol in Figure 1

The other way of applying the Fiat-Shamir method is to set the challenge as the hash of the concatenation of the public inputs, the commitment, and an arbitrary message m, i.e. $Chl \leftarrow H(Pub \parallel Cmt \parallel m)$. This will give us a *signature scheme*. The resulting signature from applying such a transform to the protocol in Figure 1 using the Random Oracle hash function $H : \{0,1\}^* \mapsto ChSp$, will be as in Figure 3, with the algorithms Sign and Verify for signing a message and verification of a candidate signature, respectively. Similarly, σ is a signature that can be verified publicly. The resulting signature scheme will be existentially unforgeable under chosen message attack if the original protocol is a Σ protocol [PS00a]. Security of the signature can be also proved assuming other requirements [OO98] or even weaker requirements on the protocol [AABN02]. We do not get into those details since we are not going to use those results directly.

```
Algorithm Sign(Pub, Sec, m)
  (St_P, Cmt) ← Cmt(Sec, Pub)
  Chl ← H(Pub ‖ Cmt ‖ m)
  Rsp ← Rsp(St_P, Chl)
  σ ← (Cmt, Rsp)
  return σ
```

```
Algorithm Verify(Pub, m, σ)
  Chl ← H(Pub ‖ Cmt ‖ m)
  d ← Dcd(Cmt, Chl, Rsp)
  return d
```

Figure 3: The signature scheme from applying Fiat-Shamir to the protocol in Figure 1

The term *signature of knowledge* has been used in the literature for a transformed proof of knowledge via the Fiat-Shamir transform, dating back to Camenisch and Stadler's work on group signatures [CS97a]. Let us also use the terms signature of knowledge (SoK) for both the NIPoK and Sign algorithms and the term *verification of*

knowledge (VoK) for both the NIVoK and Verify algorithms, resulting from applying Fiat-Shamir transform to a Σ protocol as mentioned above. Assuming the original protocol to be $\mathsf{PoK}\{Sec : \mathsf{Rel}(Pub, Sec)\}$, we denote the corresponding SoK and VoK by

$$\mathsf{SoK}\{Sec : \mathsf{Rel}(Pub, Sec)\} \triangleq \mathsf{NIPoK}(Pub, Sec) \quad \Big| \quad \mathsf{SoK}\{Sec : \mathsf{Rel}(Pub, Sec)\}(m) \triangleq \mathsf{Sign}(Pub, Sec, m)$$
$$\mathsf{VoK}\{Sec : \mathsf{Rel}(Pub, Sec)\}(\pi) \triangleq \mathsf{NIVoK}(Pub, \pi) \quad \Big| \quad \mathsf{VoK}\{Sec : \mathsf{Rel}(Pub, Sec)\}(m, \sigma) \triangleq \mathsf{Verify}(Pub, m, \sigma)\,.$$

2.5 On Public-Private Key Pairs

Key pairs are usually generated via a *key generation* algorithm KeyGen that takes a security parameter as input and outputs the key pair. It must be hard to compute the secret key corresponding to a given public key. We call the hard problem of computing the secret key for a given public key for a key pair the *underlying problem* of that key pair. Each public key is an *instance* of the underlying problem and the corresponding secret key is the corresponding *solution*. If key pairs are poly-time verifiable, i.e. one can efficiently verify if a given secret key corresponds to a given public key, the key generation algorithm KeyGen defines an NP relation $KeyPair$ consisting of all the possible key pairs, i.e.

$$KeyPair_k = \{(pk, sk) : (pk, sk) \leftarrow \mathsf{KeyGen}(k)\} \quad .$$

We are interested in key pairs for which there exists a Σ protocol to prove knowledge of a secret key corresponding to a given public key. Let us call the set of these key pairs $\mathbb{K}$. A Σ protocol for a key pair in $\mathbb{K}$, omitting the security parameter (where it is clear from the context), can be shown as

$$\mathsf{PoK}\{sk : \mathsf{KeyPair}(pk, sk)\} \quad .$$

Some key pairs that have Σ protocols as above are listed in Appendix C.1. These include popular key pairs like the ones of the GQ identification scheme, discrete-log-like key pairs, and key pairs of the RSA cryptosystem. We will use the term *key type* to refer to these different types of keys. For instance, we denote the keys for the GQ identification scheme by the term 'GQ-type key pairs'.

Note that (Ins, Sol), (Pub, Sec), and (pk, sk) are three ways of showing the same thing, i.e. a member of an NP relation, depending on how we are looking at the pair. We will use this intuition later to interchange notation between NP problems, proofs of knowledge and key pairs.

3 Defining the Class $\mathbb{C}$ of Signatures

Let $\mathsf{SS} = \mathsf{SS}.(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a provably-secure (standard) signature scheme. Security of the scheme, i.e. its existential unforgeability under chosen message attacks (EUF-CMA) [GMR88], is based on the hardness of an *underlying* problem denoted here by P_{SS}. Let us also denote by $PKSp$ and MSp the *public key space* (i.e. the set of all possible public keys) and the *message space* of a standard signature scheme, respectively. We define a class $\mathbb{C}$ of standard signature schemes as follows.

Definition. $\mathbb{C}$ is the set of all signature schemes SS for which there exists a pair of algorithms, Convert and Retrieve, where Convert gets the public key pk, a message m, and a valid signature σ on the message as input and *converts* the signature to a pair $\tilde{\sigma} = (\tilde{\sigma}_{\mathrm{aux}}, \tilde{\sigma}_{\mathrm{pre}})$ called *converted signature* as follows:

$$\tilde{\sigma} = (\tilde{\sigma}_{\mathrm{aux}}, \tilde{\sigma}_{\mathrm{pre}}) \leftarrow \mathsf{Convert}(pk, m, \sigma) \quad , \text{ such that:}$$

- there exists an algorithm AuxSim such that for every $pk \in PKSp$ and $m \in MSp$ the output of $\mathsf{AuxSim}(pk, m)$ is (information-theoretically) indistinguishable from $\tilde{\sigma}_{\mathrm{aux}}$,

- there exists an algorithm $\mathsf{Compute}$ that on input the public key pk, a message m, and $\tilde{\sigma}_{\text{aux}}$ *computes* a description of a *one-way function* $f(\cdot)$ and an I in the *range* of f, such that I is the image of $\tilde{\sigma}_{\text{pre}}$ under the one-way function f, i.e. for a converted signature the output of the following algorithm is `true`.

$$\begin{array}{l}\texttt{Algorithm } \mathsf{Valid}\,(pk, m, \tilde{\sigma}) \\ \quad (f, I) \leftarrow \mathsf{Compute}\,(pk, m, \tilde{\sigma}_{\text{aux}}) \\ \quad d \leftarrow (f\,(\tilde{\sigma}_{\text{pre}}) = I) \\ \quad \texttt{return } d\end{array}$$

- there exists a Σ protocol for proof of knowledge of a $Sec = \tilde{\sigma}_{\text{pre}}$ corresponding to a $Pub = (pk, m, \tilde{\sigma}_{\text{aux}})$ such that $\tilde{\sigma}$ is valid with respect to pk and m, i.e. there exist a Σ protocol for the following proof of knowledge

$$\mathsf{PoK}\,\{\tilde{\sigma}_{\text{pre}} : \mathsf{Valid}\,(pk, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}))\} \quad ,$$

and for any candidate converted signature satisfying $\mathsf{Valid}\,(pk, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}))$, a valid signature on the message m can be *retrieved* via the $\mathsf{Retrieve}$ algorithm as follows:

$$\sigma \leftarrow \mathsf{Retrieve}\,(pk, m, \tilde{\sigma}) \quad .$$

The definition basically requires that a signature can be bidirectionally transformed to a pair consisting of a publically-simulatable value $\tilde{\sigma}_{\text{aux}}$ and a pre-image of I under the one-way function f, where both I and f are determined by pk, m, and $\tilde{\sigma}_{\text{aux}}$ and knowledge of the pre-image $\tilde{\sigma}_{\text{pre}}$ can be proved via a HVZK-PoK protocol with SpS property. This fact enables a holder of a signature to efficiently prove knowledge of a signature on a known message to a verifier by first converting it and then revealing the simulatable part of the converted signature which enables the verifier to determine I and f. Finally, the protocol for proof of knowledge of the pre-image of I under f is carried out by the two parties. A similar property for signature schemes have been observed before in the literature, often referred to as the *reduction* of the proof of knowledge of a signature to a proof of knowledge of a pre-image under a one-way function (see e.g. [ASW00, CD00, GMY06]). Note that the fact that any NP relation has a Σ protocol [CDV06] provides protocols for proving knowledge of a signature for any signature scheme, but such protocols are not necessarily efficient enough for practice. We observe that even existence of a Σ protocol for a converted version of the signature is enough for our constructions. Such a protocol is not necessarily HVZK with respect to the signature since it reveals $\tilde{\sigma}_{\text{aux}}$.

We actually need another requirement on the signature scheme to be able to prove our schemes secure. We require that in the security proof of the signature scheme, two separate algorithms be identifiable: an algorithm that given an instance Ins of the underlying problem $\mathsf{P_{SS}}$, is able to *simulate* for the adversary a public key and signatures on the messages of its choice, and a second algorithm that given a forgery by the adversary (resp. two forgeries on the same message for schemes with proof of unforgeability based on the Forking Lemma), is able to *calculate* the solution Sol to the problem instance. We will call these two algorithms Sim and Cal, respectively. Since, this is true for all the conventional signature schemes, we do not see it as a real requirement. For more on this extra requirement, see Appendix C.2.

The definition above may seem too restricting, but many of the signature schemes in use today fall in the class $\mathbb{C}$. Examples of such schemes are RSA-FDH [BR96], Schnorr [Sch91], Modified ElGamal [PS00a], BLS [BLS01], BB [BB04], Cramer-Shoup [CS00], Camenisch-Lysyanskaya-02 [CL02], and Camenisch-Lysyanskaya-04 [CL04] signatures. Appendix C.3 lists the corresponding algorithms for the above signatures and shows why each of them belong to $\mathbb{C}$.

4 Universal Designated Verifier Signatures

In this section, we first review the definitions of the UDVS scheme and its security. Then we propose our generic construction of UDVS schemes from any signature scheme in $\mathbb{C}$ and prove it secure.

4.1 Definition

As mentioned before, a UDVS can be seen as a signature scheme with some extra functionality: a holder of a signature can designate the signature to a particular verifier, using the verifier's public key. In this sense, a UDVS can be described by adding some extra algorithms to the ones needed for description of the underlying signature scheme. Here, we briefly recall the definitions from Steinfeld et al. [SBWP03]. A UDVS is described by eight algorithms: a *Common Parameter Generation* algorithm CPGen that on input 1^k, where k is the security parameter, outputs a string consisting of common parameters cp publicly shared by all users, two *Signer (resp. Verifier) Key Generation* algorithms $\mathsf{SKeyGen}$ (resp. $\mathsf{VKeyGen}$) that on input a common parameter string cp, output a secret/public key-pair $(sk_{\mathrm{s}}, pk_{\mathrm{s}})$ (resp. $(sk_{\mathrm{v}}, pk_{\mathrm{v}})$) for the signer (resp. verifier), *Signing* and *Public Verification* algorithms Sign and PVer, where the former on input a signing secret key sk_{s} and a message m, outputs a signer's publicly-verifiable (PV) signature σ and the latter on input signer's public key pk_{s} and message/PV-signature pair (m, σ), outputs a boolean verification decision, *Designation* and *Designated Verification* algorithms Desig and DVer, where the former on input a signer's public key pk_{s}, a verifier's public key pk_{v}, and a message/PV-signature pair (m, σ), outputs a designated-verifier (DV) signature $\hat{\sigma}$ and the latter on input a signer's public key pk_{s}, a verifier's secret key sk_{v}, and a message/DV-signature pair $(m, \hat{\sigma})$, outputs a boolean verification decision, and finally a *Verifier Key-Registration* $\mathsf{VKeyReg}$ algorithm, which is a protocol between a *Key Registration Authority* (KRA) and a verifier to register verifier's public key.

4.2 Security

As mentioned before, Steinfeld et al. identified two security requirements for UDVS schemes: *DV-unforgeability* and *non-transferability privacy*. We also consider a third requirement proposed by Lipmaa et al. called *non-delegatability*. Informally and intuitively, *DV-unforgeability* captures the inability of the adversary to forge designated signatures for new messages, even if it can have signatures on chosen messages and can verify chosen pairs of messages and designated signatures, *non-transferability privacy* captures the inability of the designated verifier to produce evidence to convince a third party that the message has actually been signed by the signer, and finally *non-delegatability* captures the inability of all but either the signature holder or the designated verifier to generate designated signatures and hence the signature holder and designated verifier's inability to delegate their ability to generate designated signatures without revealing their corresponding secrets, i.e. the signature or the designated verifier secret key.

DV-Unforgeability. The formal definition of DV-Unforgeability for UDVS schemes comes in Appendix D. Non-Transferability Privacy. Steinfeld et al. have formalized this property in detail and proposed a definition capturing the fact that possessing a designated signature does not add to the computational ability of the designated verifier [SBWP03]. In their formalization, they require that whatever a designated verifier who has been given a designated signature can leak to a third party (even at the expense of disclosing his secret key), he would have been able to leak without the designated signature. One can easily see that if designated signatures are simulatable by the verifier himself then a designated signature adds no computational ability to the verifier and thus, without going into details of the formal definition for non-transferability privacy, we will state and use the following lemma to prove our schemes secure.

Lemma 1 *A scheme* UDVS *achieves* perfect *non-transferability privacy if there exists an efficient forgery algorithm* Forge*, s.t. for any pairs* $(sk_{\mathrm{s}}, pk_{\mathrm{s}})$ *and* $(sk_{\mathrm{v}}, pk_{\mathrm{v}})$ *generated through key generation algorithms of* UDVS *and for any message* m*, the following two random variables have the same distribution:*

$$\mathsf{Forge}\,(pk_{\mathrm{s}}, sk_{\mathrm{v}}, pk_{\mathrm{v}}, m) \quad \textit{and} \quad \mathsf{Desig}\,(pk_{\mathrm{s}}, pk_{\mathrm{v}}, m, \mathsf{Sign}\,(sk_{\mathrm{s}}, m)) \ .$$

Other flavors of non-transferability privacy, i.e. *statistical* and *computational* non-transferability privacy can be analogously achieved by requiring the two distributions to be statistically or computationally indistinguishable, respectively. Note that there are two main differences between this lemma and Lemma 1 in [SBWP03, p. 531]. Firstly, their lemma is biconditional, but ours is not. We are only using the direction that is pretty obvious. However, our lemma is a generalization of that direction. They only state their lemma for deterministic designated signatures, but our lemma is stated for the general (possibly probabilistic) case.

NON-DELEGATABILITY. Lipmaa et al have defined the non-delegatability property for designated-verifier signatures [LWB05]. As they mention, their definition of κ-non-delegatability basically requires the designated signature to be a non-interactive *proof of knowledge* with knowledge error κ of the signer's or the designated verifier's secret key, as per definition of [BG92]. The reason behind such a definition is to guarantee that only the signer or the designated verifier are able to produce a designated signature, thus preventing them from being able to delegate their ability without revealing their secret key. Their definition can be easily extended to the UDVS case. Since in a UDVS scheme, we want only a person who holds a signature or the designated verifier to be able to produce a designated signature, the definition can be analogously extended to the UDVS case as follows. κ-non-delegatability for UDVS schemes requires the designated signature to be a non-interactive proof of knowledge of a signature or the designated verifier's secret key, with knowledge error κ.

A nice observation by Cramer et al. [CDM00, p. 359], that will help us simplify the non-delegatability proofs for our constructions, is that a three-move public-coin protocol with SpS property and challenge space $ChSp$ is a proof of knowledge with knowledge error $\kappa = |ChSp|^{-1}$. The non-interactive version of this observation can be easily seen to hold in the Random Oracle Model using the Forking Lemma. That is, a Fiat-Shamir non-interactive proof of knowledge (i.e. our NIPoK) with SpS property and challenge space $ChSp$ is a non-interactive κ-proof of knowledge in the the Random Oracle Model with knowledge error $\kappa = |ChSp|^{-1}$. Based on these observations, we propose the following lemma:

Lemma 2 *A scheme* UDVS *is κ-non-delegatable if a designated signature is a Fiat-Shamir non-interactive proof of knowledge of a signature or the secret key of the verifier, with SpS property and $|ChSp| \geq \frac{1}{\kappa}$.*

4.3 Construction of UDVS Schemes from Standard Signatures

We show how to extend any signature scheme in class $\mathbb{C}$ to a universal designated verifier signature, by combining it with a key type for the verifier in $\mathbb{K}$. We use the building blocks we introduced before, namely proofs of disjunctive knowledge and the Fiat-Shamir transforms to construct our UDVS schemes. As mentioned before, our construction has the distinctive property that the verifier's key pair type can be chosen *independently* from the choice of the signer's signature. Our construction works for any combination of a signature in class $\mathbb{C}$ and a verifier key pair type in $\mathbb{K}$. Let $\mathsf{SS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a standard signature scheme in class $\mathbb{C}$ and KT be a verifier-chosen key type in $\mathbb{K}$. Denoting the signer- and verifier-related variables respectively by s and v indexes, the construction can be shown as follows:

- CPGen gets as input 1^k, where k is the security parameter, returns $cp = 1^k$ as the common parameter. The signer and the verifiers choose their own signature scheme and key pair types, respectively, i.e.

$$\mathsf{GUDVS}.(\mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{PVer}) \stackrel{\triangle}{=} \mathsf{SS}.(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}) \quad \text{and} \quad \mathsf{VKeyGen} \stackrel{\triangle}{=} \mathsf{KeyGen} \ .$$

- To designate, the signature-holder first converts the signature and then constructs a signature of disjunctive knowledge of $\tilde{\sigma}_{\mathrm{pre}}$ or the verifier's secret key. The DV-signature is a pair consisting of $\tilde{\sigma}_{\mathrm{aux}}$ and this signature of knowledge, i.e.

$$\begin{array}{l}
\texttt{Algorithm } \mathsf{GUDVS.Desig}\,(pk_{\mathrm{s}}, pk_{\mathrm{v}}, m, \sigma) \\
\quad (\tilde{\sigma}_{\mathrm{aux}}, \tilde{\sigma}_{\mathrm{pre}}) \leftarrow \mathsf{Convert}\,(pk_{\mathrm{s}}, m, \sigma) \\
\quad \delta \leftarrow \mathsf{SoK}\,\{(\tilde{\sigma}_{\mathrm{pre}} \vee sk_{\mathrm{v}}) : \mathsf{Valid}\,(pk_{\mathrm{s}}, m, (\tilde{\sigma}_{\mathrm{aux}}, \tilde{\sigma}_{\mathrm{pre}}))\,, \mathsf{Pair}\,(pk_{\mathrm{v}}, sk_{\mathrm{v}})\} \\
\quad \hat{\sigma} \leftarrow (\tilde{\sigma}_{\mathrm{aux}}, \delta) \\
\quad \texttt{return } \hat{\sigma}
\end{array}$$

- To verify the DV-signature, one verifies the validity of the signature of knowledge δ according to the message, the public keys of the signer and the verifier, and the value $\tilde{\sigma}_{\mathrm{aux}}$ provided, i.e.

$$\begin{array}{l}
\texttt{Algorithm } \mathsf{GUDVS.DVer}\,(pk_{\mathrm{s}}, pk_{\mathrm{v}}, m, \hat{\sigma}) \\
\quad d \leftarrow \mathsf{VoK}\,\{(\tilde{\sigma}_{\mathrm{pre}} \vee sk_{\mathrm{v}}) : \mathsf{Valid}\,(pk_{\mathrm{s}}, m, (\tilde{\sigma}_{\mathrm{aux}}, \tilde{\sigma}_{\mathrm{pre}}))\,, \mathsf{Pair}\,(pk_{\mathrm{v}}, sk_{\mathrm{v}})\}\,(\delta) \\
\quad \texttt{return } d
\end{array}$$

An example for an all-RSA-based scheme, which combines RSA-FDH signature and GQ-type verifier keys is described in Appendix E.

Note that the designated verification algorithm in our construction is *public*, since from the verifier key pair, only the public key pk_{v} is sufficient to run the GUDVS.DVer algorithm. In fact, the definition of UDVS schemes does not require that designated verification should be only executable by the verifier and for instance, the $\mathsf{SchUDVS}_2$ and RSAUDVS schemes from [SWP04] have public designated verification schemes. However, some authors have proposed a notion of *privacy of signer identity* in UDVS schemes that requires an only-verifier-executable designated verification [LV04]. In the same work, the authors show that if the designated signature is *encrypted* by the designator under an IND-CCA encryption and then sent to the verifier, then it will be verifiable only by the verifier and the scheme will preserve privacy of signer identity.

4.4 Security Analysis for the Construction

DV-Unforgeability. We use the *Forking Lemma* to prove DV-Unforgeability of our generic UDVS construction. The Forking Lemma was originally proposed by Pointcheval and Stern [PS00a]. Recently, Bellare and Neven proposed a general version of the Forking Lemma in [BN06]. We use the results and formulations from the latter in our proof. For completeness, we have transcribed the general Forking Lemma that we use in Appendix G. Basically, our *SoK*-type constructions guarantees the ability to extract a signature or the verifier's secret key from a DV-forger through forking. The extracted signature or secret key is later used to solve the underlying problem of the signature scheme or that of the verifier key pair, respectively. Thus, given a successful DV-forger, we will be able to solve at least one of the above underlying problems and we have the following theorem.

Theorem 1 *Let* SS *be a standard signature in* $\mathbb{C}$ *and* P_{SS} *be its underlying problem. Also, let KT be a key type in* $\mathbb{K}$ *and* P_{KT} *be its underlying problem. The construction* GUDVS *based on the combination of the signature* SS *and the verifier key-type KT is DV-unforgeable if* P_{SS} *and* P_{KT} *are both hard.*

The proof is given in Appendix A.

Non-Transferability Privacy. Non-transferability privacy for our generic UDVS schemes is due to the very concept behind our construction. Our designated signatures consist of publicly-simulatable values of $\tilde{\sigma}_{\mathrm{aux}}$ and *witness indistinguishable* signatures of knowledge of a valid converted signature or the verifier's secret key, both forgeable by the designated verifier himself indistinguishably from the real designated signatures. To forge a designated signature, the verifier will first simulate $\tilde{\sigma}_{\mathrm{aux}}$ via the algorithm AuxSim and then, similar to the prover, he will be able to construct a non-interactive proof of disjunctive knowledge of $\tilde{\sigma}_{\mathrm{pre}}$ or the verifier's secret key (knowing the latter, of course). The forged designated signature will be consisting of the simulated $\tilde{\sigma}_{\mathrm{aux}}$ along with this signature of knowledge, i.e. we have the following forge algorithm:

```
Algorithm GUDVS.Forge (pk_s, sk_v, pk_v, m)
  σ̃_aux ← AuxSim (pk_s, m)
  δ ← SoK {(σ̃_pre ∨ sk_v) : Valid (pk_s, m, (σ̃_aux, σ̃_pre)) , Pair (pk_v, sk_v)}
  σ̂ ← (σ̃_aux, δ)
  return σ̂
```

AuxSim's ability to simulate $\tilde{\sigma}_{\mathrm{aux}}$ and witness indistinguishability of the signature of knowledge will together imply that the output of the algorithm GUDVS.Forge is indistinguishable from real designated signatures. The existence of AuxSim and a Σ protocol for proof of knowledge of a converted signature is guaranteed if SS belongs to $\mathbb{C}$. Furthermore, the existence of a Σ protocol for proof of knowledge of the verifier secret key is guaranteed if KT belongs to $\mathbb{K}$. Thus, GUDVS.Forge will be successful in forging designated signatures for any combination of a signature in $\mathbb{C}$ and a verifier key type in $\mathbb{K}$. Combining this with Lemma 1, we will have the following theorem.

Theorem 2 *The construction* GUDVS *achieves non-transferability privacy for any combination of a signature in* $\mathbb{C}$ *and a verifier key type in* $\mathbb{K}$.

NON-DELEGATABILITY. As one can notice, the very design of our UDVS construction is naturally geared to provide non-delegatability through the use of signatures of knowledge. However, to meet the requirements of Lemma 2, we must first prove that a designated signature in our scheme is a signatures of knowledge of a *signature* or the secret key of the verifier with SpS property. All we know now is that a designated signature in our scheme consists of a $\tilde{\sigma}_{\text{aux}}$ and a signature of knowledge of $\tilde{\sigma}_{\text{pre}}$ or the secret keys of the verifier with both HVZK and SpS properties.

One can easily see that a designated signature $(\tilde{\sigma}_{\text{aux}}, \delta)$ as a signature of knowledge has the SpS property in the Random Oracle Model. The reason is that two designated signatures with the same first-move message (i.e. Random Oracle query, which includes $\tilde{\sigma}_{\text{aux}}$ along with the commitment) and different challenges (i.e. Random Oracle responses) will provide two δs with the same commitment and different challenges, which in turn, will give us the secret, i.e. $\tilde{\sigma}_{\text{pre}}$ or sk_{v}. If the former is given, then one can retrieve a valid signature by running the Retrieve algorithm on input $(\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})$. Thus, two designated signatures with the same Random Oracle query and different Random Oracle responses will give us a signature or the verifier's secret key. Hence, the designated signature will have the SpS property as well and by Lemma 2 we will have the following theorem:

Theorem 3 *The construction* GUDVS *is κ-non-delegatable for any combination of a signature in $\mathbb{C}$ and a verifier key type in $\mathbb{K}$ for which $|ChSp| \geq \frac{1}{\kappa}$.*

Note that although a designated signature is an HVZK signature of knowledge of a $\tilde{\sigma}_{\text{pre}}$ or the verifier's public key, it is *not* an HVZK signature of knowledge of a *signature* or the verifier's public key, since it reveals $\tilde{\sigma}_{\text{aux}}$ which might include some information about the signature. However, Lemma 2 does not require the designated signature to have the HVZK property.

4.5 Further Constructions

Our constructions can be easily extended to *universal multi-designated-verifier signatures*, where a signature is designated to more than one verifier. This can be done by setting the designated signature to be a one-out-of-$(n+1)$ disjunctive signature of knowledge of the (converted) signature and the secret keys of the n verifiers. Again, these schemes allow the signer and the verifiers to choose their settings independently, thus the verifiers might have different types of keys.

Similarly, the constructions can be extended to designate more than one signatures at once. This will come handy in situations where a user wishes to show more than one certificate to a verifier and does not want the verifier to be able to convince a third party of the validity of her certificate. For instance, consider a situation where a user must show at least k out of n cetificates to a verifier to obtain a service from the verifier. The user will construct the designated signature by constructing a $(k+1)$-out-of-$(n+1)$ signature of knowledge of the n (converted) signatures and the secret key of the verifier. This construction offers an extra privacy property in that the verifier will not be able to distinguish which k cetificates out of n the user has after seeing a designated signature.

4.6 Comparison

We compare our constructions with those of [SBWP03, SWP04] as benchmarks. For the comparisons to make sense, instances of our generic constructions have been chosen which match the signature scheme and verifier key type of the benchmark schemes. Similar to the comparisons in [SWP04], we count the cost of computing a product $a^x \cdot b^y \cdot c^z$ and $O(\alpha)$ low exponent exponentiations both as equivalent to a single exponentiation. We also use the same typical parameters for lengths of members of different groups in use, namely 1.024 kb for DL groups and RSA modules and 0.16 kb for $ChSp$. To make comparisons even easier, we only consider the dominant term for the costs of computation assuming that a pairing (pair.) $\succ$ an exponentiation (exp.) $\succ$ a multiplication (mult.) $\succ$ an addition, with "$\succ$" standing for "costs (much) more than". We also observe that designation of a certain certificate can be performed in two phases: before choosing the designated verifier and

after that. Analogously, designation computations can be carried out in two phases, which we denote respectively by *off-line* and *on-line* phases. An interesting property of our constructions is that cost of the on-line phase of designation is relatively very low (one multiplication). This makes our constructions desirable for the systems in which certificates are often needed to be verified by (and hence designated to) multiple different verifiers. Table 2 summarizes our comparisons, with "Typ." and "NDeleg." standing for "Typical" and "Non-Delegatability", respectively and comparatively more desirable values in bold. As the table shows, our schemes generally have more (yet comparable) costs of off-line designation and designated verification and result in longer designated signatures. However, our schemes have less on-line designation cost and achieve provable non-delegatabilty. Note that our schemes are also (almost) generic and provide the desirable property of *signer-verifier setting independence*, as mentioned before.

Table 2: Comparison of Steinfeld et al's schemes with their corresponding GUDVS counterparts

Scheme	Hard problem	Desig cost		DVer cost	Typ. $\hat{\sigma}$ length	NDeleg.
		off-line	on-line			
DVSBM [SBWP03]	BDH	**none**	1 pair.	**1 pair.**	**1.0 kb**	✗
GUDVS (BLS+DL)	**CDH**	2 pair.	**1 mult.**	2 pair.	5.3 kb	✓
$\mathsf{SchUDVS}_1$ [SWP04]	SDH	**1 exp.**	1 exp.	**1 exp.**	2.0 kb	✗
$\mathsf{SchUDVS}_2$ [SWP04]	**DL**	2 exp.	1 exp.	2 exp.	**1.5 kb**	?
GUDVS (Schnorr+DL)	**DL**	4 exp.	**1 mult.**	3 exp.	5.3 kb	✓
RSAUDVS [SWP04]	**RSA**	**1 exp.**	2 exp.	2 exp.	11.6 kb	?
GUDVS (RSA-FDH+DL)	RSA & DL	2 exp.	**1 mult.**	2 exp.	**4.3 kb**	✓

Note that, as a side effect of using the Forking Lemma for proof of security, our security reductions are not *tight*. It is possible to get tighter security results using the method proposed by Fischlin [Fis05] instead of Fiat-Shamir transform to make the interactive proofs non-interactive. However Fischlin's method will produce much longer signatures of knowledge.

5 Identity-based Signatures

In this section, we first review the definitions of the IBS scheme and its security. Then we propose our generic construction of IBS schemes from any signature scheme in $\mathbb{C}$ and prove it secure.

5.1 Definition and Security

Identity-based cryptosystems were proposed by Shamir [Sha84] to overcome the problem of lack of public-key infrastructure which the public-key cryptosystems face. In such systems, public-key certificates are no longer needed, and the identities of the users are used as their public keys. However, users lose their ability to construct their own secret keys by themselves and must depend on a *key-generation center (KGC)* to provide them with their respective private keys.

An identity-based signature is a tuple of four algorithms as follows: a *master key generation* algorithm MKeyGen, which on input a security parameter k outputs a pair of master secret key and master public key (msk, mpk), a *user key generation* algorithm UKeyGen, which on input a master secret key msk and a user identity id, outputs a user secret key usk, a *signing* algorithm Sign, which on input a user secret key usk and a message m, outputs a signature σ on the message, and finally a *verification* algorithm Verify, which on input a master public key mpk, a user identity id, and a pair (m, σ), outputs a binary decision indicating whether or not σ is a valid signature on m with respect to mpk and id.

We will use Bellare and Neven's definition for the security of an IBS scheme [BNN04] against existential unforgeability under a chosen message and identity attack, denoted by ID-EUF-CMA-attack. This definition comes in Appendix D.

5.2 Generic Construction of IBS and Its Security

In this section we show how to extend any signature in $\mathbb{C}$ to an IBS scheme. The idea is to use the key pair generated for the signature scheme as the master key pair and use the signing algorithm as the user key generation in the following way: to generate a user secret key for an identity, the identity is signed and the signature on the identity is given to the user as the user secret key. Now, the user is able to prove her identity, since she can prove knowledge of a converted signature on her identity. The Fiat-Shamir transform can be used to transform this proof into a signature scheme. The resulting signature would be an identity-based signature.

The concrete description of the generic construction is as follows. Suppose that the standard signature $\mathsf{SS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is in $\mathbb{C}$. The generic IBS scheme GIBS is constructed as follows:

To generate a master key pair, the KCG runs the key generation algorithm of the signature scheme and outputs the generated public/secret key pair as the master public/secret key pair. To generate a user key pair, the KCG simply signs the identity of the user using his master secret key and outputs the generated signature coupled with the master public key and the identity of the user as the user secret key, i.e.

$\mathtt{Algorithm}\ \mathsf{GIBS.MKeyGen}\,(k)$
$\quad (msk, mpk) \leftarrow \mathsf{SS.KeyGen}\,(k)$
$\quad \mathtt{return}\ \ (msk, mpk)$

$\mathtt{Algorithm}\ \mathsf{GIBS.UKeyGen}\,(msk, id)$
$\quad \sigma \leftarrow \mathsf{SS.Sign}\,(msk, id)$
$\quad usk \leftarrow (mpk, id, \sigma)$
$\quad \mathtt{return}\ usk$

An identity-based signature is constructed as a signature of knowledge of KGC's signature on the identity of the signer by first converting corresponding conversion algorithm on input σ (which is contained in the user secret key of the signer) to obtain $(\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})$. Then she constructs a proof of knowledge of $\tilde{\sigma}_{\text{pre}}$ and transforms it into a signature of knowledge on m via the Fiat-Shamir transform. The signature is a pair consisting of $\tilde{\sigma}_{\text{aux}}$ and this signature of knowledge. To verify an identity-based signature σ, one verifies the validity of the signature of knowledge δ according to the identity of the signer, the master public key, and the value $\tilde{\sigma}_{\text{aux}}$ provided, i.e.

$\mathtt{Algorithm}\ \mathsf{GIBS.Sign}\,(usk, m)$
$\quad (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}) \leftarrow \mathsf{Convert}\,(mpk, id, \sigma)$
$\quad \delta \leftarrow \mathsf{SoK}\,\{\tilde{\sigma}_{\text{pre}} : \mathsf{Valid}\,(mpk, id, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}))\}\,(m)$
$\quad \sigma \leftarrow (\tilde{\sigma}_{\text{aux}}, \delta)$
$\quad \mathtt{return}\ \sigma$

$\mathtt{Algorithm}\ \mathsf{IBS.Verify}\,(mpk, id, m, \sigma)$
$\quad d \leftarrow \mathsf{VoK}\,\{\tilde{\sigma}_{\text{pre}} : \mathsf{Valid}\,(mpk, id, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}))\}\,(m, \delta)$
$\quad \mathtt{return}\ d$

As mentioned before, this construction is a generalized version of Kurosawa and Heng's construction [KH04]. They require a stronger requirement on their signature schemes. It is also worth mentioning similarities between the idea behind Kurosawa and Heng's and our constructions and that of Naor's observation on how to transform any identity-based encryption to a standard signature scheme [BF01, p. 226]: in both, user secret keys are seen as the signature of the KGC on the user identity and vice versa. Our constructions can be seen as the other way of Naor's observation, i.e. from the non-identity-based world to the identity-based world. A possible result of combining the two ideas is the construction of identity-based signatures from identity-based encryptions.

We propose the following theorem for the security of our construction. A sketch of the proof is given in Appendix B.

Theorem 4 *Let* SS *be a standard signature in* $\mathbb{C}$ *and* P_{SS} *be its underlying problem. The construction* GIBS *based on the signature* SS *is* ID-EUF-CMA*-secure if* P_{SS} *is hard.*

5.3 Further Constructions

We observe that the above construction of generic IBS schemes has kind of a *nesting* property, meaning that if one extends the definition of class $\mathbb{C}$ to identity-based signature schemes, then the construction GIBS will belong

to the class $\mathbb{C}$ itself. This is due the fact that a GIBS signature in the form $\sigma = (\tilde{\sigma}_{\text{aux}}, (Cmt, Rsp))$ can be converted to the converted signature bellow:

$$\tilde{\tilde{\sigma}} = \left(\tilde{\tilde{\sigma}}_{\text{aux}}, \tilde{\tilde{\sigma}}_{\text{pre}}\right) = \left(\left(\tilde{\sigma}_{\text{aux}}, Cmt\right), Rsp\right) \quad .$$

For all the signatures listed in Appendix C.3, knowledge of Rsp can be proved via a Σ protocol. Hence, for all the constructions of IBS schemes from these signatures, the GIBS can be *nested* in the way that an identity based signer can act as a new KGC for a new user. This enables construction of *hierarchical* identity-based signature schemes [GS02].

An extension of our GIBS construction that stems from the nesting property is the construction of *identity-based universal designated verifier signatures* (IBUDVS) from any signature in $\mathbb{C}$. In such a scheme, a designator wishes to designate a certificate signed by an identity-based signer and the designated verifier is also identity-based. The designated verifier's secret key is a signature on his identity by the KGC. To designate, the designator will simply construct a disjunctive proof of knowledge of (a converted version of) her certificate *or* (a converted version of) the verifier's secret key. Proofs of security of the scheme can be constructed by combining the ideas used to prove the generic UDVS and IBS schemes secure.

Another possible extension of the GIBS schemes is the construction of *identity-based ring signatures* from any signature scheme in $\mathbb{C}$. To generate a ring signature, the signer will construct a one-out-of-n signature of knowledge of the n user secret keys in the chosen ring, where each user secret key is a signature of the KGC on the corresponding user identity.

6 Concluding Remarks

We have proposed generic constructions of UDVS and IBS schemes for a large class of signatures. Our constructions result in schemes with comparable cost and size to those of their counterparts. Our generic UDVS constructions are provably non-delegatable and also offer a signer-verifier setting independence feature. Many IBS schemes can be seen as instances of our generic IBS construction. It is possible to use our techniques to construct generic universal multi-designated-verifier signatures, hierarchical identity-based signatures, identity-based universal designated verifier signatures, and identity-based ring signatures

Acknowledgments

The authors would like to thank Shaoquan Jiang and the anonymous reviewers of PKC '08 for fruitful discussions and comments. The first author extends his thanks to the *i*CORE Information Security Lab of the University of Calgary for hosting him during part of the work.

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*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433. Springer, 2002. 6

[ASW00] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic Fair Exchange of Digital Signatures. *Selected Areas in Communications, IEEE Journal on*, 18(4):593–610, 2000. 8

[BB04] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. In Cachin and Camenisch [CC04], pages 56–73. 2, 8, 30

[BDZ04] Feng Bao, Robert H. Deng, and Jianying Zhou, editors. *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004*, volume 2947 of *Lecture Notes in Computer Science*. Springer, 2004. 18, 19, 20

[BF01] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001. 14

[BG92] Mihir Bellare and Oded Goldreich. On Defining Proofs of Knowledge. In Brickell [Bri93], pages 390–420. 10

[BLMQ05] Paulo S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In Roy [Roy05], pages 515–532. 3

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001. 2, 8, 29

[BN06] Mihir Bellare and Gregory Neven. Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 390–399. ACM, 2006. 11, 20

[BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security Proofs for Identity-Based Identification and Signature Schemes. In Cachin and Camenisch [CC04], pages 268–286. 3, 13, 25, 32

[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*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2002. 25

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993. 3

[BR96] Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In *EUROCRYPT*, pages 399–416, 1996. 2, 8, 28, 31

[Bri93] Ernest F. Brickell, editor. *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*. Springer, 1993. 16, 19

[BSS05] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Universal Designated Verifier Signature Proof (or How to Efficiently Prove Knowledge of a Signature). In Roy [Roy05], pages 644–661. 3

[CC03] Jae Choon Cha and Jung Hee Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. In Desmedt [Des02], pages 18–30. 3

[CC04] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004. 15, 16

[CD00] Jan Camenisch and Ivan Damgård. Verifiable Encryption, Group Encryption, and Their Applications to Separable Group Signatures and Signature Sharing Schemes. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2000. 8

[CDM00] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie. Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In Imai and Zheng [IZ00], pages 354–372. 10

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994. 2, 5, 34

[CDV06] Dario Catalano, Yevgeniy Dodis, and Ivan Visconti. Mercurial Commitments: Minimal Assumptions and Efficient Constructions. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 120–144. Springer, 2006. 8

[CL02] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, 2002. 2, 8, 30, 31

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004. 2, 8, 31

[CL06] Melissa Chase and Anna Lysyanskaya. On Signatures of Knowledge. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 78–96. Springer, 2006. 4

[CM07] Jean-Sébastien Coron and Alexander May. Deterministic Polynomial-Time Equivalence of Computing the RSA Secret Key and Factoring. *Journal of Cryptology*, 20(1):39–50, 2007. 26

[Cor00] Jean-Sébastien Coron. On the Exact Security of Full Domain Hash. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000. 28

[CS97a] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997. 4, 6

[CS97b] Jan Camenisch and Markus Stadler. Proof Systems For General Statements about Discrete Logarithms. Technical Report 260, Dept. of Computer Science, ETH Zurich, Mar 1997. 5

[CS00] Ronald Cramer and Victor Shoup. Signature Schemes Based on the Strong RSA Assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000. 2, 4, 8, 30

[Des02] Yvo Desmedt, editor. *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*. Springer, 2002. 16, 17

[DKXY03] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong Key-Insulated Signature Schemes. In Desmedt [Des02], pages 130–144. 3

[ElG85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. 29

[Fis05] Marc Fischlin. Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2005. 13

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. 2, 3, 6

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure Hash-and-Sign Signatures Without the Random Oracle. In *EUROCRYPT*, pages 123–139, 1999. 4, 31

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.*, 17(2):281–308, 1988. 4, 7, 31

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, 18(1):186–208, 1989. 5

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening Zero-Knowledge Protocols Using Signatures. *J. Cryptology*, 19(2):169–209, 2006. 8

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A "Paradoxical" Indentity-Based Signature Scheme Resulting from Zero-Knowledge. In Shafi Goldwasser, editor, *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer, 1988. 2, 3, 5, 28, 30, 31

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002. 15

[GW04] Shafi Goldwasser and Erez Waisbard. Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 77–100. Springer, 2004. 3, 31

[HBSO03] Jason E. Holt, Robert W. Bradshaw, Kent E. Seamons, and Hilarie K. Orman. Hidden Credentials. In Sushil Jajodia, Pierangela Samarati, and Paul F. Syverson, editors, *WPES*, pages 1–8. ACM, 2003. 4

[HCW05] Zhenjie Huang, Kefei Chen, and Yumin Wang. Efficient Identity-Based Signatures and Blind Signatures. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *CANS*, volume 3810 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2005. 3

[Hes02] Florian Hess. Efficient Identity Based Signature Schemes Based on Pairings. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2002. 3

[HSMW06] Xinyi Huang, Willy Susilo, Yi Mu, and Wei Wu. Universal Designated Verifier Signature Without Delegatability. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 479–498. Springer, 2006. 2

[HSMZ06] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Restricted Universal Designated Verifier Signature. In Jianhua Ma, Hai Jin, Laurence Tianruo Yang, and Jeffrey J. P. Tsai, editors, *UIC*, volume 4159 of *Lecture Notes in Computer Science*, pages 874–882. Springer, 2006. 3

[IZ00] Hideki Imai and Yuliang Zheng, editors. *Public Key Cryptography, Third International Workshop on Practice and Theory in Public Key Cryptography, PKC 2000, Melbourne, Victoria, Australia, January 18-20, 2000, Proceedings*, volume 1751 of *Lecture Notes in Computer Science*. Springer, 2000. 16, 19

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. In *EUROCRYPT*, pages 143–154, 1996. 1, 2

[KH04] Kaoru Kurosawa and Swee-Huay Heng. From Digital Signature to ID-based Identification/Signature. In Bao et al. [BDZ04], pages 248–261. 3, 14

[LLP05] Yong Li, Helger Lipmaa, and Dingyi Pei. On Delegatability of Four Designated Verifier Signatures. In Sihan Qing, Wenbo Mao, Javier Lopez, and Guilin Wang, editors, *ICICS*, volume 3783 of *Lecture Notes in Computer Science*, pages 61–71. Springer, 2005. 2

[LLQ06] Fabien Laguillaumie, Benoît Libert, and Jean-Jacques Quisquater. Universal Designated Verifier Signatures Without Random Oracles or Non-black Box Assumptions. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2006. 3

[LV04] Fabien Laguillaumie and Damien Vergnaud. Designated Verifier Signatures: Anonymity and Efficient Construction from Any Bilinear Map. In Carlo Blundo and Stelvio Cimato, editors, *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2004. 11

[LW06] Jin Li and Yanming Wang. Universal Designated Verifier Ring Signature (Proof) Without Random Oracles. In Xiaobo Zhou, Oleg Sokolsky, Lu Yan, Eun-Sun Jung, Zili Shao, Yi Mu, Dong Chun Lee, Daeyoung Kim, Young-Sik Jeong, and Cheng-Zhong Xu, editors, *EUC Workshops*, volume 4097 of *Lecture Notes in Computer Science*, pages 332–341. Springer, 2006. 3

[LWB05] Helger Lipmaa, Guilin Wang, and Feng Bao. Designated Verifier Signature Schemes: Attacks, New Security Notions and a New Construction. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 459–471. Springer, 2005. 1, 2, 10

[MT05] Akihiro Mihara and Keisuke Tanaka. Universal Designated-Verifier Signature with Aggregation. In *ICITA (2)*, pages 514–519. IEEE Computer Society, 2005. 3

[NSM05] Ching Yu Ng, Willy Susilo, and Yi Mu. Universal Designated Multi Verifier Signature Schemes. In *ICPADS (2)*, pages 305–309. IEEE Computer Society, 2005. 3, 4

[Oka92] Tatsuaki Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In Brickell [Bri93], pages 31–53. 3

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 354–369. Springer, 1998. 6

[PS00a] David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. Cryptology*, 13(3):361–396, 2000. 2, 6, 8, 11, 28, 29

[PS00b] Guillaume Poupard and Jacques Stern. Short Proofs of Knowledge for Factoring. In Imai and Zheng [IZ00], pages 147–166. 26

[PS06] Kenneth G. Paterson and Jacob C. N. Schuldt. Efficient Identity-Based Signatures Secure in the Standard Model. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP*, volume 4058 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2006. 3

[Roy05] Bimal K. Roy, editor. *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*. Springer, 2005. 16

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal Designated-Verifier Signatures. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542. Springer, 2003. 1, 2, 3, 9, 12, 13, 31

[Sch91] Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *J. Cryptology*, 4(3):161–174, 1991. 2, 5, 8, 28, 29, 30, 31

[Sha84] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO*, pages 47–53, 1984. 2, 3, 13

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. *Sympoium on Cryptography and Information Security (SCIS), Okinawa, Japan*, pages 26–28, January 2000. 3

[SWP04] Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures. In Bao et al. [BDZ04], pages 86–100. 2, 3, 11, 12, 13, 31

[Ver06] Damien Vergnaud. New Extensions of Pairing-Based Signatures into Universal Designated Verifier Signatures. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 58–69. Springer, 2006. 3

[Yi03] Xun Yi. An Identity-Based Signature Scheme from the Weil Pairing. *Communications Letters, IEEE*, 7(2):76–78, 2003. 3

[ZFI05] Rui Zhang, Jun Furukawa, and Hideki Imai. Short Signature and Universal Designated Verifier Signature Without Random Oracles. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 483–498, 2005. 3

[ZSMC05] Fangguo Zhang, Willy Susilo, Yi Mu, and Xiaofeng Chen. Identity-Based Universal Designated Verifier Signatures. In Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuanshun Dai, and Laurence Tianruo Yang, editors, *EUC Workshops*, volume 3823 of *Lecture Notes in Computer Science*, pages 825–834. Springer, 2005. 3, 4

[ZSS04] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In Bao et al. [BDZ04], pages 277–290. 31

A Proof of Theorem 1

Proof. Let GUDVS be a UDVS scheme constructed via our constructions from a signature scheme SS in $\mathbb{C}$ and a verifier key pair type KT in $\mathbb{K}$. Let also the underlying hard problem of the signature scheme be P_{SS} and the underlying hard problem of the verifier key pair type be P_{KT}. Given a DV-forger A and two instances of the problems P_{SS} and P_{KT}, we will show that at least one of the problem instances can be solved.

We will show how to construct, given a DV-forger A, two *solver* algorithms Slv_{KT} and $\mathsf{Slv}_{\mathsf{SS}}$ for solving P_{KT} and P_{SS} instances, respectively. We will also show that at least one of these two strategies will succeed in solving its given instance of the problem, if the DV-forger manages to forge successfully. Given a successful DV-forger A, Slv_{KT} will succeed only if the forgery produced by the A is of a certain type which is defined by an event. We also show that $\mathsf{Slv}_{\mathsf{SS}}$ succeeds if the DV-forger A is successful and another event occurs. Furthermore, we will show that the events above cover the universe. It follows that, with the above two solvers, given a DV-forger for GUDVS scheme , at least one of them will solve the associated problem.

We will construct our solvers in a modular way. We will introduce four algorithms Sim_{KT}, $\mathsf{Sim}_{\mathsf{SS}}$, Cal_{KT}, and $\mathsf{Cal}_{\mathsf{SS}}$ and use these four algorithms along with the adversary A and the Bellare-Neven forker algorithm (see Appendix G) as modules of constructing the two solvers. We will construct the solver Slv_{KT} as follows:

- the *simulator* algorithm Sim_{KT} will run A as a subroutine, simulating the attack environment (inputs and answers to queries) for A, and obtain a DV-forgery from A,
- the *forker* algorithm Frk_{KT} will run Sim_{KT} as a subroutine, forking inputs to it, and obtain two different designated signatures from it, and
- the *solution calculator* algorithm Cal_{KT} will run Frk_{KT} as a subroutine and use the two designated signatures output by it to solve the given instance of the problems P_{KT}.

The solver $\mathsf{Slv}_{\mathsf{SS}}$ is also constructed in a similar way, using algorithms $\mathsf{Sim}_{\mathsf{SS}}$, $\mathsf{Frk}_{\mathsf{SS}}$, and $\mathsf{Cal}_{\mathsf{SS}}$. The algorithms $\mathsf{Sim}_{\mathsf{SS}}$ and $\mathsf{Cal}_{\mathsf{SS}}$, in turn, run the Sim and Cal algorithms of the signature scheme SS, respectively. These algorithms are defined in Appendixes C.2 and C.3. The forker algorithms are also based on the constructions of Bellare and Neven [BN06] as discussed in Appendix G. We will describe each module in the following and discuss how they work and lead to the proof. First we will describe Sim_{KT}, Frk_{KT}, and Cal_{KT} algorithms, which are used to construct the algorithm for solving a given P_{KT} problem instance. After a discussion on the success probability of our solver, we will proceed to introduce our second set of algorithms $\mathsf{Sim}_{\mathsf{SS}}$, $\mathsf{Frk}_{\mathsf{SS}}$, and $\mathsf{Cal}_{\mathsf{SS}}$, which are used to construct the algorithm for solving a given P_{SS} problem instance. We will denote the random oracles used in the signature scheme by $\mathcal{H}_{\mathsf{SS}}$ and the one used in the Fiat-Shamir transform to build a signature of knowledge by $\mathcal{H}_{\mathsf{FS}}$.

Let us first set some notations. One can see easily from Figures 9 and 2 and our construction that our designated signatures will be in the form $\hat{\sigma} = (\tilde{\sigma}_{\mathrm{pre}}, \delta)$, where

$$\delta = (Cmt, Rsp) = ((Cmt_{\mathrm{s}}, Cmt_{\mathrm{v}}), (Chl_{\mathrm{s}}, Chl_{\mathrm{v}}, Rsp_{\mathrm{s}}, Rsp_{\mathrm{v}})) \quad .$$

Furthermore, we have

$$Chl_{\mathrm{s}} + Chl_{\mathrm{v}} = Chl = H_{\mathsf{FS}}\left(Pub \parallel Cmt\right) = H_{\mathsf{FS}}\big(\left(Pub_{\mathrm{s}}, Pub_{\mathrm{v}}\right) \parallel \left(Cmt_{\mathrm{s}}, Cmt_{\mathrm{v}}\right)\big) \ .$$

Also note that, following our proof of knowledge notation convention, the notation

$$\delta \leftarrow \mathsf{SoK}\left\{\left(\tilde{\sigma}_{\mathrm{pre}} \vee sk_{\mathrm{v}}\right) : \mathsf{Valid}\left(pk_{\mathrm{s}}, m, \left(\tilde{\sigma}_{\mathrm{aux}}, \tilde{\sigma}_{\mathrm{pre}}\right)\right), \mathsf{Pair}\left(pk_{\mathrm{v}}, sk_{\mathrm{v}}\right)\right\}$$

implies that

$$Sec_{\mathrm{s}} = \tilde{\sigma}_{\mathrm{pre}}, \quad Sec_{\mathrm{v}} = sk_{\mathrm{v}}, \quad Pub_{\mathrm{s}} = \left(pk_{\mathrm{s}}, m, \tilde{\sigma}_{\mathrm{aux}}\right), \quad \text{and} \quad Pub_{\mathrm{v}} = pk_{\mathrm{v}} \ .$$

We will use the above notation throughout the proof.

Algorithm Sim_{KT} gets a P_{KT} instance Ins and a q-tuple $(h_1, \ldots, h_q)$ as input. It first runs the key generation algorithm of the corresponding signature to obtain a key pair $(sk_{\mathrm{s}}, pk_{\mathrm{s}})$. Then Sim_{KT} runs A with inputs pk_{s} and $pk_{\mathrm{v}} = Ins$. Note that, as we mentioned before, one can see the problem instance Ins as a public key pk_{v}, for which we are trying to find the solution Sol, i.e. corresponding secret key sk_{v}. During its run, A will ask $\mathcal{H}_{\mathsf{SS}}$, $\mathcal{H}_{\mathsf{FS}}$, and $\mathcal{S}ign$ oracle queries. Sim_{KT} simulates the answers as follows:

- answers $\mathcal{H}_{\mathsf{SS}}$ queries randomly and records the answers.
- answers $\mathcal{H}_{\mathsf{FS}}$ queries by taking elements of the q-tuple $(h_1, \ldots, h_q)$ consecutively, i.e. answers the first query with h_1, the second with h_2 and so on.
- answers $\mathcal{S}ign$ queries by running the Sign algorithm of the signature scheme. Note that the signing key sk_{s} is known to Sim_{KT}, so there is no need to simulate the signatures.

At last, A outputs a DV-forgery $(m, \hat{\sigma})$, where $\hat{\sigma} = (\tilde{\sigma}_{\mathrm{aux}}, \delta)$. Sim_{KT} checks whether or not the adversary has been successful in forging, i.e. checks whether or not the message is new and the DV-forgery is valid by running the DVer algorithm. If $(m, \hat{\sigma})$ passes both tests, denoting $\delta = \left(\left(Cmt_{\mathrm{s}}, Cmt_{\mathrm{v}}\right), \left(Chl_{\mathrm{s}}, Chl_{\mathrm{v}}, Rsp_{\mathrm{s}}, Rsp_{\mathrm{v}}\right)\right)$, Sim_{KT} looks up the *index* J s.t. $h_J = Chl_{\mathrm{s}} + Chl_{\mathrm{v}}$ and outputs $(J, (m, \hat{\sigma}))$. In the case that the adversary has not been successful or no matching index J is found, Sim_{KT} outputs $(0, \varepsilon)$.

Algorithm Frk_{KT} takes as input a P_{KT} instance Ins. It is defined as the Bellare-Neven forker algorithm in Appendix G, with input Ins and access to algorithm Sim_{KT}, i.e. using the Bellare-Neven notation

$$\mathsf{Frk}_{KT} \stackrel{\triangle}{=} \mathsf{F}_{\mathsf{Sim}_{KT}}\left(Ins\right) \ .$$

Frk_{KT} outputs either $\left(1, \left(m, \hat{\sigma}\right), \left(m', \hat{\sigma}'\right)\right)$ or $(0, \varepsilon, \varepsilon)$, depending on whether the forking has been successful or not. Note that a successful forking implies same Jth $\mathcal{H}_{\mathsf{FS}}$ oracle query and different corresponding answers. Since queries are of the form $\left(Pub_{\mathrm{s}}, Pub_{\mathrm{v}}\right) \parallel \left(Cmt_{\mathrm{s}}, Cmt_{\mathrm{v}}\right)$, where $Pub_{\mathrm{s}} = \left(pk_{\mathrm{s}}, m, \tilde{\sigma}_{\mathrm{aux}}\right)$, the message m is also part of the Jth query and thus is the same for the two runs of the forked algorithm Sim_{KT}, hence $m = m'$. Therefore, from now on, we will use m instead of m'.

Algorithm Cal_{KT} takes as input a P_{KT} instance Ins. It first runs Frk_{KT} on the same input Ins and obtains either $\left(1, \left(m, \hat{\sigma}\right), \left(m, \hat{\sigma}'\right)\right)$ or $(0, \varepsilon, \varepsilon)$. Receiving the former means that forking by Frk_{KT} has been successful, i.e. $J = J'$ and $h_J \neq h'_J$ according to the general Forking Lemma. Note that h_J and h'_J are the two responses to the Jth $\mathcal{H}_{\mathsf{FS}}$ oracle queries in the two runs of the forked algorithm Sim_{KT}. Thus $h_J = Chl = Chl_{\mathrm{s}} + Chl_{\mathrm{v}}$ and $h'_J = Chl' = Chl'_{\mathrm{s}} + Chl'_{\mathrm{v}}$. Hence we have the following event:

$$\mathbf{E} \stackrel{\triangle}{=} \left[\quad J = J' \quad \wedge \quad Chl_{\mathrm{s}} + Chl_{\mathrm{v}} \neq Chl'_{\mathrm{s}} + Chl'_{\mathrm{v}} \quad \right] \ . \tag{1}$$

Now, if $Chl_{\mathrm{v}} \neq Chl'_{\mathrm{v}}$, then Cal_{KT} will simply run the extraction algorithm for the protocol for proof of knowledge of the verifier's secret key and get a sk_{v} s.t. $\mathsf{Pair}\left(pk_{\mathrm{v}}, sk_{\mathrm{v}}\right)$. Cal_{KT} outputs $Sol = sk_{\mathrm{v}}$ as the solution to the P_{KT} problem instance Ins. If $Chl_{\mathrm{v}} = Chl'_{\mathrm{v}}$, Cal_{KT} declares failure and halts.

A graphical depiction of how modules are wired to interact in our solver is shown in Figure 4. Note that, again, random oracle queries are not shown in the figure. Let us denote by Slv_{KT} our solver, i.e. the combination of all our modules: Cal_{KT}, Frk_{KT}, and the two instances of Sim_{KT} wired together as in Figure 4.

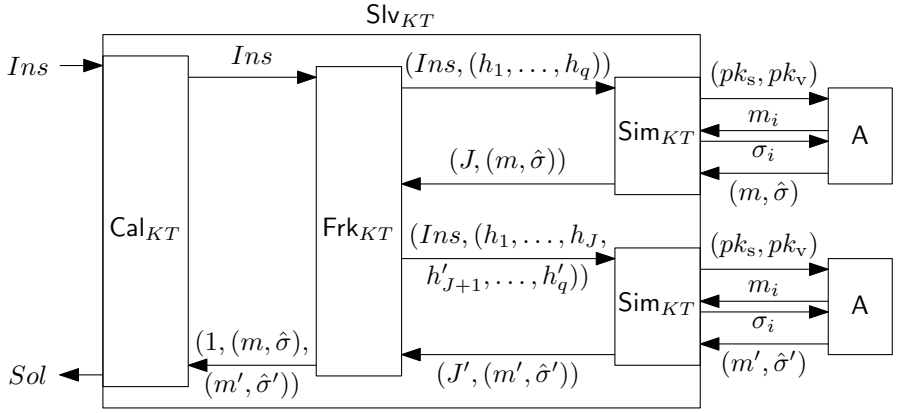

Figure 4: Mechanism of the proof

Let us calculate the probability that our solver is successful in solving the P_{KT} problem instance Ins. We define the success probabilities for Sim_{KT} and Frk_{KT} similar to acc and frk, respectively, in the general Forking Lemma (see Appendix G), i.e.

- $\mathrm{Adv}_{\mathsf{Sim}_{KT}(\mathsf{A})}(k)$ is defined as the probability that Sim_{KT}'s first output is not zero, given A, a random problem instance of size k, and random choices of $h_1, \ldots, h_q$, and
- $\mathrm{Adv}_{\mathsf{Frk}_{KT}(\mathsf{Sim}_{KT})}(k)$ is defined as the probability that Frk_{KT}'s first output is one, given Sim_{KT} and a random problem instance of size k.

Now we observe that Sim_{KT} succeeds if A succeeds in forging and the forgery uses a queried hash. We know that the probability that A succeeds without using a queried hash is at most one over the size of the challenge space. Thus we have

$$\mathrm{Adv}_{\mathsf{Sim}_{KT}(\mathsf{A})}(k) \geq \mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A}(\mathsf{GUDVS})}(k) - \frac{1}{|ChSp|} \ .$$

On the other hand, we have Bellare and Neven's General Forking Lemma which gives us a lower bound for the success probability of the forker, i.e. Frk_{KT}, based on the success probability of the simulator, i.e. Sim_{KT}. Thus we will have

$$\mathrm{Adv}_{\mathsf{Frk}_{KT}(\mathsf{Sim}_{KT})}(k) \geq \mathrm{Adv}_{\mathsf{Sim}_{KT}(\mathsf{A})}(k) \cdot \left(\frac{\mathrm{Adv}_{\mathsf{Sim}_{KT}(\mathsf{A})}(k)}{q} - \frac{1}{|ChSp|} \right) \ ,$$

where q is the maximum number of $\mathcal{H}_{\mathsf{FS}}$ queries A makes. We also see that Cal_{KT} is successful if Frk_{KT} succeeds and $Chl_\mathrm{v} \neq Chl'_\mathrm{v}$. So we get the following:

$$\mathrm{Adv}^{\mathrm{P}_{KT}}_{\mathsf{Cal}_{KT}(\mathsf{Frk}_{KT})}(k) = \Pr\left[Chl_\mathrm{v} \neq Chl'_\mathrm{v} | \mathsf{Frk}_{KT} \text{ succeeds}\right] \cdot \mathrm{Adv}_{\mathsf{Frk}_{KT}(\mathsf{Sim}_{KT})}(k) \, .$$

Combining the three equations above, and applying the fact that Frk_{KT} succeeds *iff* $\mathbf{E}$ happens, we can compute the overall probability of success of our solver in solving P_{KT} as follows:

$$\begin{aligned} \mathrm{Adv}^{\mathrm{P}_{KT}}_{\mathsf{Slv}_{KT}(\mathsf{A})}(k) \quad \geq \quad & \frac{1}{q} \cdot \Pr\left[Chl_\mathrm{v} \neq Chl'_\mathrm{v} | \mathbf{E}\right] \cdot \\ & \cdot \left(\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A}(\mathsf{GUDVS})}(k) - \frac{1}{|ChSp|} \right) \cdot \left(\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A}(\mathsf{GUDVS})}(k) - \frac{1+q}{|ChSp|} \right) \ . \end{aligned}$$

Assuming that the size of the challenge space is super-logarithmic in the security parameter and the number of queries the adversary asks is polynomially-bounded in the security parameter, we can neglect the two fractions with $|ChSp|$ as denominator and simplify the above equation as follows:

$$\mathrm{Adv}^{\mathrm{P}_{KT}}_{\mathsf{Slv}_{KT}(\mathsf{A})}(k) \geq \frac{1}{q} \cdot \Pr\left[Chl_\mathrm{v} \neq Chl'_\mathrm{v} | \mathbf{E}\right] \cdot \left[\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A}(\mathsf{GUDVS})}(k) \right]^2 \ . \tag{2}$$

Now we describe the three algorithms $\mathsf{Sim_{SS}}$, $\mathsf{Frk_{SS}}$, and $\mathsf{Cal_{SS}}$ for solving an instance Ins of the underlying problem of the signature scheme $\mathrm{P_{SS}}$. These modules are again wired together as shown in Figure 4, changing all the indexes from KT to SS.

Algorithm $\mathsf{Sim_{SS}}$ gets an instance Ins of the problem $\mathrm{P_{SS}}$ and a q-tuple $(h_1, \ldots, h_q)$ as input. It first runs the corresponding verifier key generation algorithm KeyGen to obtain a key pair $(sk_{\mathrm{v}}, pk_{\mathrm{v}})$. Then $\mathsf{Sim_{SS}}$ runs the simulator algorithm Sim of the signature scheme SS on input Ins to get a public key pk_{s} for the signature scheme. It then runs A on input $(pk_{\mathrm{s}}, pk_{\mathrm{v}})$. A will ask $\mathcal{H}_{\mathsf{SS}}$, $\mathcal{H}_{\mathsf{FS}}$, and $\mathcal{S}ign$ oracle queries. $\mathsf{Sim_{SS}}$ responds as follows:

- forwards all $\mathcal{H}_{\mathsf{SS}}$ and $\mathcal{S}ign$ oracle queries to the signature simulator Sim and relays the answers given by Sim back to A.
- answers $\mathcal{H}_{\mathsf{FS}}$ queries with the q-tuple it is provided with, i.e. answers the first query with h_1, the second with h_2 and so on.

If Sim succeeds in simulating the $\mathcal{H}_{\mathsf{SS}}$ and $\mathcal{S}ign$ oracle queries, at last A outputs a DV-forgery $(m, \hat{\sigma})$. $\mathsf{Sim_{SS}}$ checks whether or not the adversary has been successful in forging, i.e. checks whether or not the message is new and the DV-forgery is valid by running the DVer algorithm. If $(m, \hat{\sigma})$ passes both tests, $\mathsf{Sim_{SS}}$ looks up the *index* J s.t. $h_J = Chl_{\mathrm{s}} + Chl_{\mathrm{v}}$ and outputs $(J, (m, \hat{\sigma}))$. In the case that either Sim fails, the adversary fails in forging a valid forgery, or no matching index J is found, $\mathsf{Sim_{SS}}$ outputs $(0, \varepsilon)$.

Algorithm $\mathsf{Frk_{SS}}$ takes as input an instance Ins of the problem $\mathrm{P_{SS}}$. It is defined as the Bellare-Neven forker algorithm, with input Ins and access to algorithm $\mathsf{Sim_{SS}}$, i.e.

$$\mathsf{Frk_{SS}} \stackrel{\triangle}{=} \mathsf{F}_{\mathsf{Sim_{SS}}}(Ins) \quad .$$

$\mathsf{Frk_{SS}}$ outputs either $(1, (m, \hat{\sigma}), (m', \hat{\sigma}'))$ or $(0, \varepsilon, \varepsilon)$. Note that, with a similar reasoning as before, $m = m'$.

Algorithm $\mathsf{Cal_{SS}}$ takes as input an instance Ins of the problem $\mathrm{P_{SS}}$. It runs $\mathsf{Frk_{SS}}$ on the same input and obtains either $(1, (m, \hat{\sigma}), (m, \hat{\sigma}'))$ or $(0, \varepsilon, \varepsilon)$. Again, receiving the former means that forking by $\mathsf{Frk_{SS}}$ has been successful, i.e. $J = J'$ and $h_J \neq h'_J$. Hence we have the same event $\mathbf{E}$ as defined in Equation 1. Now, if $Chl_{\mathrm{s}} \neq Chl'_{\mathrm{s}}$, then $\mathsf{Cal_{SS}}$ will simply run the extraction algorithm for the protocol for proof of knowledge of $\tilde{\sigma}_{\mathrm{pre}}$ and get a $\tilde{\sigma}_{\mathrm{pre}}$ s.t. $\mathsf{Valid}\left(pk_{\mathrm{s}}, m, (\tilde{\sigma}_{\mathrm{aux}}, \tilde{\sigma}_{\mathrm{pre}})_{\mathrm{v}}\right)$. Then it runs the corresponding $\mathsf{Retrieve}$ algorithm on input $(\tilde{\sigma}_{\mathrm{aux}}, \tilde{\sigma}_{\mathrm{pre}})$ and gets a valid σ. Now, $\mathsf{Cal_{SS}}$ feeds (m, σ) to the solution calculator algorithm Cal of the signature scheme SS and gets the solution Sol for the problem instance Ins of the problem $\mathrm{P_{SS}}$ if Cal is successful. If either $Chl_{\mathrm{s}} = Chl'_{\mathrm{s}}$ or Cal fails, $\mathsf{Cal_{SS}}$ declares failure and halts.

Let us calculate the probability that our solver is successful in solving the $\mathrm{P_{SS}}$ problem instance Ins. We can define the success probability for $\mathsf{Sim_{SS}}$ and $\mathsf{Frk_{SS}}$ similar to that of Sim_{KT} and Frk_{KT}. Notice that $\mathsf{Sim_{SS}}$ succeeds if Sim succeeds in simulating, A succeeds in forging, and the forgery uses a queried hash. Thus, with similar reasonings as before, the success probability of $\mathsf{Sim_{SS}}$ can be finally written as

$$\mathrm{Adv}_{\mathsf{Sim_{SS}(A)}}(k) \geq \mathrm{Adv}_{\mathsf{Sim(A)}}(k) \cdot \left(\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A(GUDVS)}}(k) - \frac{1}{|ChSp|} \right) \quad .$$

Furthermore, for the success probability of the forker algorithm $\mathsf{Frk_{SS}}$, a similar equation to the one we had in the first part of the proof holds, i.e.

$$\mathrm{Adv}_{\mathsf{Frk_{SS}(Sim_{SS})}}(k) \geq \mathrm{Adv}_{\mathsf{Sim_{SS}(A)}}(k) \cdot \left(\frac{\mathrm{Adv}_{\mathsf{Sim_{SS}(A)}}(k)}{q} - \frac{1}{|ChSp|} \right) \quad ,$$

where q is the maximum number of $\mathcal{H}_{\mathsf{FS}}$ queries made by A. We also see that $\mathsf{Cal_{SS}}$ is successful if $\mathsf{Frk_{SS}}$ succeeds in forking, $Chl_{\mathrm{v}} \neq Chl'_{\mathrm{v}}$, and Cal succeeds in solving the problem instance Ins. Now let us define the following event:

$$\mathbf{F} \stackrel{\triangle}{=} [\quad \mathsf{Frk_{SS}} \text{ succeeds} \quad \wedge \quad Chl_{\mathrm{s}} \neq Chl'_{\mathrm{s}} \quad] \quad .$$

In case of event $\mathbf{F}$, a valid signature σ on the message m can be computed. The probability of obtaining such a signature can be written as

$$\Pr[\mathbf{F}] = \Pr[Chl_{\mathrm{s}} \neq Chl'_{\mathrm{s}} | \mathsf{Frk_{SS}} \text{ succeeds}] \cdot \mathrm{Adv}_{\mathsf{Frk_{SS}(Sim_{SS})}}(k) \quad .$$

Now, using the notation defined in Appendix C.2, for non-FL-based signatures, we will have the following:

$$\mathrm{Adv}^{\mathrm{P_{SS}}}_{\mathsf{Cal_{SS}(Frk_{SS})}}(k) \geq \mathrm{Adv}_{\mathsf{Cal(Sim)}}(k) \cdot \Pr[\mathbf{F}] \ .$$

Combining the above equations and with a similar reasoning that lead us to Equation 2 plus the fact that $\mathbf{E}$ happens *iff* $\mathsf{Frk_{SS}}$ succeeds, we get the following for overall probability of success of our solver in solving $\mathrm{P_{SS}}$ for non-FL-based signatures:

$$\begin{aligned}\mathrm{Adv}^{\mathrm{P_{SS}}}_{\mathsf{Slv_{SS}}}(k) \quad \geq \quad & \frac{1}{q} \cdot \mathrm{Adv}_{\mathsf{Cal(Sim)}} \cdot \Pr[Chl_{\mathrm{s}} \neq Chl'_{\mathrm{s}} | \mathbf{E}] \cdot \left[\mathrm{Adv}_{\mathsf{Sim(A)}}(k)\right]^2 \cdot \\ & \cdot \left(\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A(GUDVS)}}(k) - \frac{1}{|ChSp|}\right) \cdot \left(\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A(GUDVS)}}(k) - \frac{1+q}{|ChSp| \cdot \mathrm{Adv}_{\mathsf{Sim(A)}}(k)}\right) \ ,\end{aligned}$$

where we also have exploited the fact that $\mathrm{Adv}_{\mathsf{Sim(A)}}(k) \leq 1$ to change the last numerator from $\mathrm{Adv}_{\mathsf{Sim(A)}}(k) + q$ to $1 + q$. Similarly, assuming that the size of the challenge space is super-logarithmic, the number of queries the adversary asks is polynomially-bounded, and $\mathrm{Adv}_{\mathsf{Sim(A)}}(k)$ is noticeable in the security parameter, we can simplify the above equation as follows:

$$\mathrm{Adv}^{\mathrm{P_{SS}}}_{\mathsf{Slv_{SS}}}(k) \geq \frac{1}{q} \cdot \mathrm{Adv}_{\mathsf{Cal(Sim)}} \cdot \Pr[Chl_{\mathrm{s}} \neq Chl'_{\mathrm{s}} | \mathbf{E}] \cdot \left[\mathrm{Adv}_{\mathsf{Sim(A)}}(k)\right]^2 \cdot \left[\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A(GUDVS)}}(k)\right]^2 \ . \tag{3}$$

Combining Equations 2 and 3 and applying the fact that

$$\Pr[Chl_{\mathrm{v}} \neq Chl'_{\mathrm{v}} | \mathbf{E}] + \Pr[Chl_{\mathrm{s}} \neq Chl'_{\mathrm{s}} | \mathbf{E}] \geq 1 \ ,$$

we will get the following result for non-FL-based schemes:

$$\mathrm{Adv}^{\mathrm{P}_{KT}}_{\mathsf{Sim}_{KT}(\mathsf{A})}(k) + \frac{1}{\mathrm{Adv}_{\mathsf{Cal(Sim)}} \cdot \left[\mathrm{Adv}_{\mathsf{Sim(A)}}(k)\right]^2} \cdot \mathrm{Adv}^{\mathrm{P_{SS}}}_{\mathsf{Slv_{SS}(A)}}(k) \geq \frac{1}{q} \cdot \left[\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A(GUDVS)}}(k)\right]^2 \ .$$

Thus, as long as the adversary A has a good advantage of forging, we will be able to solve at least one of the problem instances of P_{KT} or $\mathrm{P_{SS}}$ with a good probability. This completes the proof for non-FL-based signatures.

On the other hand, for the FL-based signatures, since another forking is performed to get two valid signatures, we get the following:

$$\mathrm{Adv}^{\mathrm{P_{SS}}}_{\mathsf{Cal_{SS}(Frk_{SS})}}(k) \geq \mathrm{Adv}_{\mathsf{Cal(Sim)}}(k) \cdot \Pr[\mathbf{F}] \cdot \left(\frac{\Pr[\mathbf{F}]}{q} - \frac{1}{|R_{H_{\mathsf{SS}}}|}\right) \ ,$$

where $R_{H_{\mathsf{SS}}}$ is the range of the hash function H_{SS}. Again with a similar reasoning, assuming that the size of the challenge space and the size of $R_{H_{\mathsf{SS}}}$ are both super-logarithmic, the number of queries the adversary asks is polynomially-bounded, and $\mathrm{Adv}_{\mathsf{Sim(A)}}(k)$ is noticeable in the security parameter, we will get the following final result for overall probability of success of our solver in solving $\mathrm{P_{SS}}$ for FL-based signatures:

$$\mathrm{Adv}^{\mathrm{P_{SS}}}_{\mathsf{Slv_{SS}}}(k) \geq \frac{1}{q^3} \cdot \mathrm{Adv}_{\mathsf{Cal(Sim)}} \cdot \left(\Pr[Chl_{\mathrm{s}} \neq Chl'_{\mathrm{s}} | \mathbf{E}]\right)^2 \cdot \left[\mathrm{Adv}_{\mathsf{Sim(A)}}(k)\right]^4 \cdot \left[\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A(GUDVS)}}(k)\right]^4 \ . \tag{4}$$

Similarly, combining Equations 2 and 4, we will get the following result for FL-based schemes:

$$\mathrm{Adv}^{\mathrm{P}_{KT}}_{\mathsf{Sim}_{KT}(\mathsf{A})}(k) + \frac{\sqrt{q}}{\sqrt{\mathrm{Adv}_{\mathsf{Cal(Sim)}}} \cdot \left[\mathrm{Adv}_{\mathsf{Sim(A)}}(k)\right]^2} \cdot \sqrt{\mathrm{Adv}^{\mathrm{P_{SS}}}_{\mathsf{Slv_{SS}(A)}}(k)} \geq \frac{1}{q} \cdot \left[\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{A(GUDVS)}}(k)\right]^2 \ ,$$

which again, guarantees a lower band for the probability that our solvers are able to solve at least one of the two instances of respectively the prroblems P_{KT} and $\mathrm{P_{SS}}$. This completes the proof for FL-based signatures. □

B Proof Sketch of Theorem 4

Proof sketch. We will prove that the interactive version of our IBS scheme, denoted by GIBI, is an identity-based identification scheme secure against impersonation under passive attacks (IMP-PA in the sense of [BNN04]). This will complete the proof since Bellare et al. have shown that any IMP-PA-secure IBI is transformed via Fiat-Shamir to a ID-EUF-CMA-secure IBS scheme [BNN04].

To prove IMP-PA security we need to be able to respond to two types of oracle queries: corruption oracle and conversation oracle queries. For the former, a user secret key for the given identity must be simulated and for the latter, a transcript of the interaction between a user with a given identity and a verifier. User secret keys can be simulated via the simulation algorithm for the signature scheme, since user secret keys are simply signatures on user identities. Transcripts of the interaction between a user with a given identity and a verifier can be simulated via first simulating the $\tilde{\sigma}_{\text{aux}}$ and then simulating a transcript for the proof of knowledge of the $\tilde{\sigma}_{\text{pre}}$ corresponding to the master public key, the identity, and $\tilde{\sigma}_{\text{aux}}$.

At last, the successful impersonator can be used to extract two transcripts with the same $\tilde{\sigma}_{\text{aux}}$ and commitment message and two different challenges and responses to them. This will allow first computing the $\tilde{\sigma}_{\text{pre}}$ and then, knowing both $\tilde{\sigma}_{\text{aux}}$ and $\tilde{\sigma}_{\text{pre}}$, computing a forgery for the signature scheme which, in turn, will be given to the solution calculator algorithm to compute the solution to the given instance of the underlying problem P_{SS}.

Given an instance of the underlying problem P_{SS}, we will run the Sim algorithm on this input. Sim will give us a pk that we will relay to the adversary as the master public key. The adversary then will start to ask two types of oracle queries: corruption oracle queries and conversation oracle queries. On a corruption oracle query id, we will forward id as a signing query to Sim and get the signature σ on it and then forward it along with the master public key and the input id as the response to the query id (i.e. the user secret key corresponding to id) to the adversary. On a conversation query id, we will run the AuxSim algorithm on input the master public key and id and get a simulated $\tilde{\sigma}_{\text{aux}}$. Then we will run the TrSim algorithm for the protocol for proof of knowledge of $\tilde{\sigma}_{\text{pre}}$ on input $(mpk, id, \tilde{\sigma}_{\text{aux}})$ to get a transcript $Tr = (Cmt, Chl, Rsp)$ for that protocol. Then we will give the adversary the conversation $((Cmt, \tilde{\sigma}_{\text{aux}}), Chl, Rsp)$ as the response to the query id. For the signature schemes that use a random oracle in their construction, the adversary will ask $\mathcal{H}_{\mathsf{SS}}$ queries as well. These queries are also relayed to Sim algorithm and the response is relayed back to the adversary.

At last, the adversary decides that the first phase is over and outputs a target identity id^*. We will keep answering the queries as before in the second phase. The adversary will be able to prove knowledge of the user secret key corresponding to id^* at this stage. Rewinding the adversary and asking for a new challenge will give us two transcripts with the same commitment and $\tilde{\sigma}^*_{\text{aux}}$ and different challenges and their respective responses. We will be able to extract a $\tilde{\sigma}^*_{\text{pre}}$ corresponding to $\tilde{\sigma}^*_{\text{aux}}$ from the same commitment, different challenges, and their respective responses then, and at last run the $\mathsf{Retrieve}$ algorithm on input the master public key, id^*, and the pair $\left(\tilde{\sigma}^*_{\text{aux}}, \tilde{\sigma}^*_{\text{pre}}\right)$ to get a signature σ^* on the identity id^*. We will finally run the Cal algorithm on input σ^* to get the solution to the problem instance.

Let us compute the probability that we will be successful in solving the underlying problem instance. Let us denote the probability that we are able to successfully simulate the environment for the adversary and the adversary will give us a *suitable* forgery by acc. With a similar reasoning to the proof of Theorem 1, we will get

$$acc\,(k) \geq \mathrm{Adv}_{\mathsf{Sim(A)}}\,(k) \cdot \mathrm{Adv}^{\text{IMP-PA}}_{\mathsf{GIBI,A}}\,(k) \quad .$$

Now, applying the Reset Lemma of Bellare and Palacio [BP02] we will get the success probability of computing two suitable transcripts as follows

$$res\,(k) \geq \left(acc\,(k) - \frac{1}{|ChSp|}\right)^2 \quad .$$

Furthermore, again with a similar reasoning to the proof of Theorem 1, we will be able to calculate the probability that our solution calculator
$mathsfCal_{SS}$ will be successful in solving the instance of the problem P_{SS} as the following for non-FL-based schemes:

$$\mathrm{Adv}^{\mathrm{P}_{\mathsf{SS}}}_{\mathsf{Cal}_{\mathsf{SS}}}\,(k) \geq \mathrm{Adv}_{\mathsf{Cal(Sim)}} \cdot res\,(k) \quad .$$

Combining the above equations, we will get the following final result for the success probability of our solver Slv of P_{SS} problem instances for non-FL-based schemes:

$$\mathrm{Adv}^{\mathrm{P}_{\mathsf{SS}}}_{\mathsf{Slv}}(k) \geq \mathrm{Adv}_{\mathsf{Cal(Sim)}} \cdot \left(\mathrm{Adv}_{\mathsf{Sim(A)}}(k) \cdot \mathrm{Adv}^{\text{IMP-PA}}_{\mathsf{A(GIBI)}}(k) - \frac{1}{|ChSp|} \right)^2 .$$

This completes the proof for non-FL-based signature schemes.

Furthermore, for FL-based signatures, the probability that the solution calculator will be successful in solving the instance of the problem P_{SS} can be written as the following:

$$\mathrm{Adv}^{\mathrm{P}_{\mathsf{SS}}}_{\mathsf{Cal}_{\mathsf{SS}}}(k) \geq \mathrm{Adv}_{\mathsf{Cal(Sim)}} \cdot \Big(res(k) \Big)^2 .$$

Thus the overall success probability of the solver will be calculated as follows for FL-based signature schemes:

$$\mathrm{Adv}^{\mathrm{P}_{\mathsf{SS}}}_{\mathsf{Slv}}(k) \geq \mathrm{Adv}_{\mathsf{Cal(Sim)}} \cdot \Big(\mathrm{Adv}_{\mathsf{Sim(A)}}(k) \cdot \mathrm{Adv}^{\text{IMP-PA}}_{\mathsf{A(GIBI)}}(k) - \frac{1}{|ChSp|} \Big)^4 .$$

This completes the proof for FL-based signatures. □

C More on Classes $\mathbb{K}$ and $\mathbb{C}$

C.1 Some Key Types in the Class $\mathbb{K}$

There are quite a few different types of key pairs used in cryptographic schemes. Three of the simplest and most popular types are RSA-type, GQ-type, and DL-type key pairs. The key generation algorithms for these types of keys are shown in Figure 5. The RSAGen and DLGen algorithms are respectively the *prime exponent RSA parameter generator* and the *DL parameter generator* algorithms that generate system parameters with respect to the security parameter taken as input.

`Algorithm` $\mathsf{RSAKeyGen}(k)$	`Algorithm` $\mathsf{GQKeyGen}(k)$	`Algorithm` $\mathsf{DLKeyGen}(k)$
$(N, e, d) \leftarrow \mathsf{RSAGen}(k)$	$(N, e, d) \leftarrow \mathsf{RSAGen}(k)$	$(p, g) \leftarrow \mathsf{DLGen}(k)$
$sk \leftarrow d$	$sk \xleftarrow{\$} \mathbb{Z}^*_N; \quad X \xleftarrow{N} sk^e$	$sk \xleftarrow{\$} \mathbb{Z}_p; \quad X \leftarrow g^{sk}$
$pk \leftarrow (N, e, d)$	$pk \leftarrow (N, e, X)$	$pk \leftarrow (p, g, X)$
`return` (pk, sk)	`return` (pk, sk)	`return` (pk, sk)

Figure 5: GQ- and DL-type key generation algorithms

The underlying problem of the RSA-type keys is finding the private exponent d of an RSA system corresponding to the values (N, e, d). The authors are not aware of any direct Σ protocols for proof of knowledge of the private exponent d corresponding to (N, e, d). However, the results of Coron and May [CM07] shows that the knowledge of the private exponent is equivalent to the knowledge of the factorization of N. Thus, instead of proving knowledge of d, one can use the existing Σ protocols for proof of knowledge of the factorization of N, for example the protocols by Poupard and Stern [PS00b]. Therefore, RSA keys belong to $\mathbb{K}$.

The underlying problem of GQ and DL key types are the RSA and DL problems, respectively. Knowledge of the secret key corresponding to a public key for these two types of keys can be proved via GQ and Schnorr protocols, respectively, which are both Σ protocols. So these two types of keys also belong to $\mathbb{K}$.

C.2 On Simulatability of Signature Schemes

We require that there exists a pair of algorithms, Sim and Cal, such that given an instance Ins of the underlying hard problem P_{SS}, Sim is able to *simulate* a public key for the signature scheme and signatures for arbitrary chosen messages with a noticeable probability, and given a pair (resp. two pairs) consisting of a new message and a signature (resp. two signatures) on the message, valid with respect to the simulated public key, Cal is able to *calculate* a solution Sol to the problem instance with a noticeable probability.

Intuitively, this property requires that it is possible to simulate the public key and signatures for chosen messages for the signature scheme, without knowledge of the secret key, with a sufficiently good probability, in a way that a forgery enables us to solve an instance of a hard problem. This simulation might take place in the Random Oracle Model though. Proofs of unforgeability for most of the signature schemes, are constructed in a folklore standard way by first simulating the attack environment for the adversary and then using the adversary's forgery to solve a hard problem. These two are the algorithms we are looking for. Note that for a proof in the ROM, the simulator must also answer A's random oracle queries as well as its signing oracle queries.

Now consider two types of signatures depending on whether or not their security proof is based on the Forking Lemma. If the security proof of SS is *not* based on Forking Lemma (*non-FL-based signature* from now on), then only one forgery is enough for the Cal algorithm to compute the solution Sol. However, if the security proof of SS *is* based on Forking Lemma (*FL-based signature* from now on), then Cal will need two signatures on the same message to be able to calculate the solution to the hard problem. Let us denote by by $\mathrm{Adv}_{\mathsf{Sim(A)}}(k)$ the probability that the Sim succeeds in simulating the attack environment for A and gets a *suitable* forgery (definition of suitable is case-dependent). Let us also denote by $\mathrm{Adv}_{\mathsf{Cal(Sim)}}(k)$ the probability that given one (respectively two for FL-based schemes) valid signature(s) on a message, Cal succeeds in computing the solution Sol for the problem instance Ins given to Sim.

A depiction of the mechanism of the proof for these two types of schemes is shown in Figure 6. Note that random oracle queries are not shown in this figure. As one can follow the order of events in Figure 6 from top to bottom, for a non-FL-based scheme, first the problem instance Ins is given to Cal as input. The public key pk and answers σ_i to signing oracle queries m_i are then simulated by Sim. The forgery (m, σ) which is output by the adversary A is then used by Cal to calculate a solution Sol for the problem instance. For FL-based schemes, a forker algorithm Frk is introduced which runs the simulator and the adversary two times, ordering the simulator to use different values as responses to the adversary's random oracle queries each time. Then the two signatures are given to the Cal that calculates and outputs Sol. For more details, see the General Forking Lemma in Appendix G.

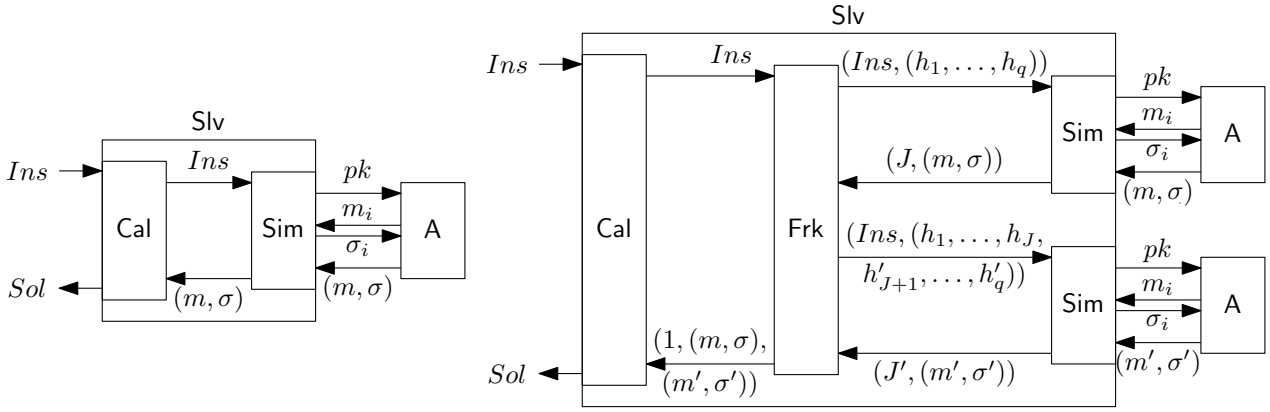

Figure 6: Mechanism of Proofs of Unforgeability for Non-FL-Based (left) and FL-Based Signatures (right)

Let us also denote by Slv the combination of Cal and Sim for the case of non-FL-based signatures and the combination of Cal, Frk, and the two instances of Sim for the case of FL-based signatures. Using the notation

defined above, we will have the following results respectively for the non-FL-based and FL-based schemes:

$$\begin{aligned}\mathrm{Adv}^{\mathrm{Pss}}_{\mathsf{Slv}}(k) &\geq \mathrm{Adv}_{\mathsf{Cal(Sim)}}(k)\cdot\mathrm{Adv}_{\mathsf{Sim(A)}}(k), \quad \text{and}\\ \mathrm{Adv}^{\mathrm{Pss}}_{\mathsf{Slv}}(k) &\geq \mathrm{Adv}_{\mathsf{Cal(Sim)}}(k)\cdot\mathrm{Adv}_{\mathsf{Sim(A)}}(k)\cdot\left(\mathrm{Adv}_{\mathsf{Sim(A)}}(k)-\frac{1}{|ChSp|}\right) .\end{aligned}$$

C.3 Some Signatures in the Class $\mathbb{C}$

RSA-FDH Signature: The *Full-Domain Hash RSA* signature scheme was proposed and proved secure by Bellare and Rogaway [BR96]. The key pairs are of the forms $pk = (N, e)$ and $sk = d$, where $ed = 1 \bmod \varphi(N)$ and N is an RSA modulus. A valid signature σ satisfies the verification equation $\sigma^e = H(m) \bmod N$. The signature is the pre-image itself and no auxiliary information $\tilde{\sigma}_{\mathrm{aux}}$ is required to carry out the proof, thus we have the following algorithms:

$$(\varepsilon, \sigma) \leftarrow \mathsf{Convert}(pk, m, \sigma) \quad \text{and} \quad \tilde{\sigma}_{\mathrm{pre}} \leftarrow \mathsf{Retrieve}(pk, m, \tilde{\sigma}) .$$

Simulation of $\tilde{\sigma}_{\mathrm{aux}} = \varepsilon$ is trivial. The verification equation suggests the following one-way function and image:

$$f(x) = x^e \bmod N \quad \text{and} \quad I = H(m) .$$

Hence, $\tilde{\sigma}_{\mathrm{pre}} = \sigma$ is the eth RSA root of I and knowledge of $\tilde{\sigma}_{\mathrm{pre}}$ can be proved via the GQ protocol [GQ88] which is a Σ protocol. Thus, RSA-FDH has the first property.

One can easily see that the second property also holds for RSA-FDH. The security proof by Coron [Cor00] can be easily seen to have two separable parts as bellow: given an RSA problem instance $Ins = (N, e, X)$, the simulator simulates the public key as $pk_{\mathrm{s}} = (N, e, X)$, answers to hash queries of m_i as either $X \cdot r_i^e \bmod N$ or $r_i^e \bmod N$ for a random r_i, and answers to sign queries on m_i as r_i. Given a forgery (m^*, σ^*), the simulator first finds the corresponding i so that $m_i = m^*$ and then checks whether or not m^* hash query has been answered by $X \cdot r_i^e \bmod N$ or not and sends the forgery to the solution calculator if positive. The solution calculator calculates $Sol = \sigma^*/r_i \bmod N$ as the solution to Ins. According to [Cor00] the above simulator will be successful in simulating and getting a suitable forgery with probability $\frac{1}{exp(1)\cdot q_s}$ and the solution calculator will be successful in solving the problem instance with probability 1, given a suitable forgery.

Schnorr Signature: Schnorr proposed the scheme for use in smart cards [Sch91] and Pointcheval and Stern proved the scheme secure [PS00a]. The key pairs are of the forms $pk = (p, q, g, h)$ and $sk = x$, where $h = g^x$. A valid signature is of the form $\sigma = (c, z)$ for a random $c \in \mathbb{Z}_q$ s.t. the verification equation $c = H(g^z \cdot h^{-c}, m)$ is satisfied. Signatures can be converted and retrieved as follows:

$$\left(g^z \cdot h^{-c}, z\right) \leftarrow \mathsf{Convert}(pk, m, \sigma) \quad \text{and} \quad \left(H(\tilde{\sigma}_{\mathrm{aux}}, m), \tilde{\sigma}_{\mathrm{pre}}\right) \leftarrow \mathsf{Retrieve}(pk, m, \tilde{\sigma}) .$$

Since c is chosen randomly, $\tilde{\sigma}_{\mathrm{aux}} = g^z \cdot h^{-c}$ is uniformly distributed and can be simulated by just picking a uniformly random element of $\mathbb{Z}_q$. The verification equation can be rewritten as

$$g^{\tilde{\sigma}_{\mathrm{pre}}} = \tilde{\sigma}_{\mathrm{aux}} \cdot h^{H(\tilde{\sigma}_{\mathrm{aux}}, m)}, \quad \text{where} \quad \tilde{\sigma}_{\mathrm{pre}} = z .$$

Therefore, we will have the following one-way function and image:

$$f(x) = g^x \quad \text{and} \quad I = \tilde{\sigma}_{\mathrm{aux}} \cdot h^{H(\tilde{\sigma}_{\mathrm{aux}}, m)} .$$

Hence, $\tilde{\sigma}_{\mathrm{pre}}$ is the discrete logarithm of I in base g and knowledge of $\tilde{\sigma}_{\mathrm{pre}}$ can be proved via the Schnorr protocol [Sch91] which is a Σ protocol. Thus, the first property is satisfied.

The second property can also easily be seen to hold for Schnorr. Since Schnorr is also a signature of knowledge constructed via Fiat-Shamir transform, signatures can be easily simulated in the Random Oracle Model. Given an instance $pk = (p, q, g, h)$, one simulates a signature on m_i by picking two random elements c and z in $\mathbb{Z}_q$ and sets $\sigma_i = (c, z)$ and also answers hash oracle queries consistent with $c = H(g^z \cdot h^{-c}, m)$. Using the Forking Lemma, one can get two forgeries on the same message from an adversary and given two forgeries on the same

message, the solution calculator can easily compute the discrete logarithm of h in base g. This observation is basically the same as that of Pointcheval and Stern on the simulatability of the Schnorr signatures [PS00a, Section 3.2.2]. The probability of the simulation success and that of the solving DL given two signatures on the same message are both 1.

MODIFIED ELGAMAL SIGNATURE: ElGamal signature scheme was proposed by ElGamal [ElG85]. A slightly-modified version was proposed and proved secure by Pointcheval and Stern [PS00a]. The key pairs are of the forms $pk = (p, g, h)$ and $sk = x$, where $h = g^x$. A signature is of the form $\sigma = (r, s)$ for a random $r \in \mathbb{Z}_{p-1}^*$ s.t. the verification equatioon $g^{H(m,r)} = h^r r^s$ is satisfied. Signatures can be converted and retrieved as follows:

$$(r, s) \leftarrow \mathsf{Convert}\,(pk, m, \sigma) \quad \text{and} \quad (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}) \leftarrow \mathsf{Retrieve}\,(pk, m, \tilde{\sigma}) \ .$$

Since $\tilde{\sigma}_{\text{aux}} = r$ is uniformly distributed, it can be simulated by just picking a uniformly random element of $\mathbb{Z}_{p-1}^*$. The verification equation can be rewritten as

$$r^{\tilde{\sigma}_{\text{pre}}} = g^{H(m,\tilde{\sigma}_{\text{aux}})} / h^{\tilde{\sigma}_{\text{aux}}}, \quad \text{where} \quad \tilde{\sigma}_{\text{pre}} = s \ .$$

Therefore, we will have the following one-way function and image:

$$f\,(x) = \tilde{\sigma}_{\text{aux}}^x \quad \text{and} \quad I = g^{H(m,\tilde{\sigma}_{\text{aux}})} / h^{\tilde{\sigma}_{\text{aux}}} \ .$$

Hence, $\tilde{\sigma}_{\text{pre}} = s$ is the discrete logarithm of I in base r and knowledge of $\tilde{\sigma}_{\text{pre}}$ can be proved via the Schnorr protocol [Sch91] which is a Σ protocol. Thus, Modified ElGamal has the first property.

The results of Pointcheval and Stern show that the second property also holds for Modified ElGamal. They have proved that For *α-hard* prime numbers, the signer can be simulated with an indistinguishable distribution [PS00a, Lemma 6] and that given two signatures on the same message the solution calculator can find a solution for the discrete logarithm problem instance with α-hard prime modulus in polynomial time [PS00a, Theorem 6]. These two results show that Modified ElGamal has the second property.

BLS SIGNATURE: The BLS signature was proposed and proved secure by Boneh et al. [BLS01]. The key pairs are of the forms $pk = (q, g, e, y)$ and $sk = x$, where $y = g^x$. A valid signature σ satisfies the equation $e\,(\sigma, g) = e\,(H\,(m)\,, y)$. Signatures can be converted and retrieved as follows:

$$(\sigma^z, z) \leftarrow \mathsf{Convert}\,(pk, m, \sigma) \ \text{ where } z \xleftarrow{\$} \mathbb{Z}_q^* \quad \text{and} \quad \tilde{\sigma}_{\text{aux}}^{1/\tilde{\sigma}_{\text{pre}}} \leftarrow \mathsf{Retrieve}\,(pk, m, \tilde{\sigma}) \ .$$

Since z is chosen randomly, $\tilde{\sigma}_{\text{aux}} = \sigma^z$ is uniformly distributed and can be simulated by just picking a uniformly random element of $\mathbb{Z}_q^*$. The verification equation can be rewritten as

$$e\,(H\,(m)\,, y)^{\tilde{\sigma}_{\text{pre}}} = e\,(\tilde{\sigma}_{\text{aux}}, g)\,, \quad \text{where} \quad \tilde{\sigma}_{\text{pre}} = z \ .$$

Therefore, we will have the following one-way function and image:

$$f\,(x) = e\,(H\,(m)\,, y)^x \quad \text{and} \quad I = e\,(\tilde{\sigma}_{\text{aux}}, g) \ .$$

Hence, $\tilde{\sigma}_{\text{pre}}$ is the discrete logarithm of I in base $e\,(H\,(m)\,, y)$ and knowledge of $\tilde{\sigma}_{\text{pre}}$ can be proved via the Schnorr protocol [Sch91] which is a Σ protocol. Thus, the first property holds for BLS.

The proposed proof of unforgeability by Boneh et al. shows that the second property also holds for BLS. Basically, given a CDH problem instance $Ins = (q, e, g, X, \bar{g})$, the simulator simulates the public key as $pk_{\text{s}} = (q, e, g, X)$, answers to hash queries of m_i as either either $\bar{g} \cdot g^{r_i}$ or g^{r_i} for a random r_i, and answers to sign queries on m_i as X^{r_i}. Given a forgery (m^*, σ^*), the solution calculator first finds the corresponding i so that $m_i = m^*$ and then checks whether or not m^* hash query has been answered by $\bar{g} \cdot g^{r_i} \bmod N$ or not and sends the forgery to the solution calculator if positive. The solution calculator calculates $Sol = \sigma^* / X^{r_i} \bmod N$ as the solution to Ins. According to [BLS01] the above simulator will be successful in simulating and getting a suitable forgery with probability $\frac{1}{2 \cdot exp(1) \cdot q_{\text{s}}}$ and the solution calculator will be successful in solving the problem instance with probability 1, given a suitable forgery.

BB Signature: The BB signature was proposed and proved secure by Boneh and Boyen [BB04]. The key pairs are of the forms $pk = (q, g, e, u_1, u_2)$ and $sk = (x, y)$, where $u_1 = g^x$ and $u_2 = g^y$. A signature is of the form $\sigma = (\delta, l)$ for a random $l \in \mathbb{Z}_q^*$ that satisfies the equation $e\left(\delta, u_1 g^m u_2^l\right) = e\left(g, g\right)$. Signatures can be converted and retrieved as follows:

$$\left(\left(\delta^z, l\right), z\right) \leftarrow \mathsf{Convert}\left(pk, m, \sigma\right) \text{ where } z \xleftarrow{\$} \mathbb{Z}_q^* \quad \text{and} \quad \left(\tilde{\delta}^{1/z}, l\right) \leftarrow \mathsf{Retrieve}\left(pk, m, \left(\left(\tilde{\delta}, l\right), z\right)\right) \ .$$

Since both l and z are chosen randomly, $\tilde{\sigma}_{\text{aux}} = \left(\tilde{\delta}, l\right) = \left(\delta^z, l\right)$ is uniformly distributed and can be simulated by just picking two uniformly random elements of $\mathbb{Z}_q^*$. The verification equation can be rewritten as

$$e\left(\tilde{\delta}, u_1 g^m u_2^l\right) = e\left(g, g\right)^{\tilde{\sigma}_{\text{pre}}}, \quad \text{where} \quad \tilde{\sigma}_{\text{pre}} = z \ .$$

Therefore, we will have the following one-way function and image:

$$f\left(x\right) = e\left(g, g\right)^x \quad \text{and} \quad I = e\left(\tilde{\delta}, u_1 g^m u_2^l\right) \ .$$

Hence, $\tilde{\sigma}_{\text{pre}}$ is the discrete logarithm of I in base $e\left(g, g\right)$ and knowledge of $\tilde{\sigma}_{\text{pre}}$ can be proved via the Schnorr protocol [Sch91] which is a Σ protocol. Thus, BB has the first property. One can examine that the second property also holds for BB (see [BB04]).

Cramer-Shoup Signature: The Cramer-Shoup signature was proposed and proved secure by Cramer and Shoup [CS00]. The key pairs are of the forms $pk = (n, h, x, e')$ and $sk = (p, q)$, where $n = pq$ is an RSA modulus, h and x are random quadratic residues mod n, and e' is prime. A signature is of the form $\sigma = (e, y, y')$ for a random prime e and a random quadratic residue y'. A valid signature satisfies the equation

$$x = y^e \cdot h^{-H\left(x'\right)} \mod n, \quad \text{where} \quad x' = \left(y'\right)^{e'} \cdot h^{-H(m)} \mod n \ .$$

Signatures can be converted and retrieved trivially as follows:

$$\left(\left(e, y'\right), y\right) \leftarrow \mathsf{Convert}\left(pk, m, \sigma\right) \quad \text{and} \quad \left(e, y, y'\right) \leftarrow \mathsf{Retrieve}\left(pk, m, \left(\left(e, y'\right), y\right)\right) \ .$$

Since both e and y' are chosen randomly, $\tilde{\sigma}_{\text{aux}} = \left(e, y'\right)$ is uniformly distributed and can be simulated by just picking two uniformly random elements from the corresponding sets. The verification equation can be rewritten as

$$\tilde{\sigma}_{\text{pre}}^e = x \cdot h^{-H\left(\left(y'\right)^{e'} \cdot h^{-H(m)}\right)} \mod n, \quad \text{where} \quad \tilde{\sigma}_{\text{pre}} = y \ .$$

Therefore, we will have the following one-way function and image:

$$f\left(x\right) = x^e \quad \text{and} \quad I = x \cdot h^{-H\left(\left(y'\right)^{e'} \cdot h^{-H(m)}\right)} \ .$$

Hence, $\tilde{\sigma}_{\text{pre}}$ is the eth RSA root of I mod n and knowledge of $\tilde{\sigma}_{\text{pre}}$ can be proved via the GQ protocol [GQ88] which is a Σ protocol. Thus, Cramer-Shoup has the first property. One can examine that the second property also holds for the scheme (see [CS00]).

Camenisch-Lysyanskaya-02 Signature: The CL02 signature was proposed and proved secure by Camenisch and Lysyanskaya [CL02]. The key pairs are of the forms $pk = (n, a, b, c)$ and $sk = (p, q)$, where $n = pq$ is an RSA modulus and a, b, and c are random quadratic residues mod n. A signature is of the form $\sigma = (e, s, v)$ for a random prime e and a random s. A valid signature satisfies the equation

$$v^e = a^{H(m)} \cdot b^s \cdot c \mod n \ .$$

Signatures can be converted and retrieved trivially as follows:

$$\left(\left(e, s\right), v\right) \leftarrow \mathsf{Convert}\left(pk, m, \sigma\right) \quad \text{and} \quad \left(e, s, v\right) \leftarrow \mathsf{Retrieve}\left(pk, m, \left(\left(e, s\right), v\right)\right) \ .$$

Since both e and s are chosen randomly, $\tilde{\sigma}_{\text{aux}} = (e, s)$ is uniformly distributed and can be simulated by just picking two uniformly random elements from the corresponding sets. The verification equation suggests the following one-way function and image:

$$f(x) = x^e \quad \text{and} \quad I = a^{H(m)} \cdot b^s \cdot c \ .$$

Hence, $\tilde{\sigma}_{\text{pre}} = v$ is the eth RSA root of I mod n and knowledge of $\tilde{\sigma}_{\text{pre}}$ can be proved via the GQ protocol [GQ88] which is a Σ protocol. Thus, CL02 has the first property. One can examine that the second property also holds for the scheme (see [CL02]).

Camenisch-Lysyanskaya-04 Signature: The CL04 signature was proposed and proved secure by Camenisch and Lysyanskaya [CL04]. The key pairs are of the forms $pk = (q, G, G', g, g', e, X, Y)$ and $sk = (x, y)$, where $G = \langle g \rangle$ and $G' = \langle g' \rangle$ are two groups of prime size q, $e : G \times G \mapsto G'$ is a pairing, $X = g^x$, and $Y = g^y$. A signature is of the form $\sigma = (a, b, c)$ for a random $a \in G$ and a valid signature satisfies the equations

$$e(a, Y) = e(g, b) \quad \text{and} \quad e(X, a) \cdot [e(X, b)]^{H(m)} = e(g, c) \ .$$

Signatures can be converted and retrieved as follows:

$$((a, b, c^r), r) \leftarrow \mathsf{Convert}(pk, m, \sigma) \text{ where } r \overset{\$}{\leftarrow} \mathbb{Z}_q^* \quad \text{and} \quad \left(a, b, \tilde{c}^{1/r}\right) \leftarrow \mathsf{Retrieve}(pk, m, ((a, b, \tilde{c}), r)) \ .$$

Since r are chosen randomly, $\tilde{c} = c^r$ is a random element of G. Furthermore, a and b are both random in G with the restriction that $e(a, Y) = e(g, b)$. Thus, $\tilde{\sigma}_{\text{aux}} = (a, b, \tilde{c})$ can be simulated by just picking two uniformly random elements $z \in \mathbb{Z}_q^*$ and $\tilde{c} \in G$ and then setting $a = g^z$ and $b = Y^z$, since we will then have $e(a, Y) = e(g, b)$. The verification equations can be rewritten as

$$e(a, Y) = e(g, b) \quad \text{and} \quad \left[e(X, a) \cdot [e(X, b)]^{H(m)}\right]^r = e(g, \tilde{c}) \ .$$

Therefore, we will have the following one-way function and image:

$$f(x) = \left[e(X, a) \cdot [e(X, b)]^{H(m)}\right]^x \quad \text{and} \quad I = e(g, \tilde{c}) \ .$$

Hence, $\tilde{\sigma}_{\text{pre}} = r$ is the discrete logarithm of I in base $e(X, a) \cdot [e(X, b)]^{H(m)}$ and knowledge of $\tilde{\sigma}_{\text{pre}}$ can be proved via the Schnorr protocol [Sch91] which is a Σ protocol. Note that CL04 signatures can be randomized by just raising to a random power and if the signature is randomized before the construction of the proof then we will get a Σ protocol for proof of knowledge of a *signature* (instead of that of $\tilde{\sigma}_{\text{pre}}$). Such protocol would be similar to the protocol described in [CL04] with the slight difference that in our case, the message is also known to the verifier. Thus, CL04 has the first property. One can examine that the second property also holds for the scheme (see [CL04]).

Other Signatures: As mentioned before, Goldwasser and Waisbard's results in [GW04] show that both Goldwasser-Micali-Rivest [GMR88] and Gennaro-Halevi-Rabin [GHR99] are also in $\mathbb{C}$. Many other pairing-based schemes can also be easily seen to be in $\mathbb{C}$, for instance the signature scheme proposed in [ZSS04]. However, there exist some schemes that does not seem to belong to $\mathbb{C}$, or at least does not seem to admit to efficient protocols, e.g. the PSS signature scheme from [BR96].

D Formal Definition of Security for UDVS and IBS Schemes

DV-Unforgeability of UDVS Schemes. In the unforgeability game, as per original definition by Steinfeld et al. [SBWP03] and the strengthened version in a later work [SWP04], the adversary is given the security parameter, the signer's as well as the verifier's public keys, and oracle access to sign any message as well as to verify any pair of message and designated signature. The adversary's goal is to forge a designated signature on a new message, i.e. on a message that has not been queried to the signing oracle. Formally, an experiment is defined for a UDVS scheme UVDS and a forger F with access to the *signing* oracle $\mathcal{S}ign$ and *designated*

```
Experiment Expt^{DV-EUF-CMA}_{F(UVDS)} (k)
  cp ← CPGen(1^k) ; M ← ∅
  (sk_s, pk_s) ← SKeyGen(cp)
  (sk_v, pk_v) ← VKeyGen(cp)
  (m, σ̂) ← F(1^k, pk_s, pk_v; Sign(·), DVer(·,·))
  if (DVer(pk_s, sk_v, m, σ̂) = 1 ∧ m ∉ M)
    then return 1 else return 0
```

```
Oracle Sign(m)
  σ ← Sign(sk_s, m)
  M ← M ∪ {m}
  return σ
```

```
Oracle DVer(m, σ̂)
  return DVer(pk_s, sk_v, m, σ̂)
```

Figure 7: DV-EUF-CMA experiment and oracles

verification oracle $\mathcal{DV}er$ as in Figure 7. The advantage of F in attacking UDVS in a DV-EUF-CMA attack is defined as:

$$\mathrm{Adv}^{\text{DV-EUF-CMA}}_{\mathsf{F(UVDS)}}(k) \stackrel{\triangle}{=} \Pr\left[\mathsf{Expt}^{\text{DV-EUF-CMA}}_{\mathsf{F(UVDS)}}(k) = 1\right] .$$

A UDVS is said to be DV-EUF-CMA-*secure* if no poly-time attacker can get an advantage non-negligible in k, in a DV-EUF-CMA attack against it.

ID-Unforgeability for IBS Schemes. We recall Bellare and Neven's definition of IBS security [BNN04] against existential unforgeability under a chosen message and identity attack, denoted here by ID-EUF-CMA-security. The adversary has the ability to initialize and corrupt users beside its ability to obtain signatures on chosen messages and identities. Formally, an experiment with corresponding *initialization* oracle $\mathcal{I}nit$, *signing* oracle $\mathcal{S}ign$, and *corruption* oracle $\mathcal{C}orr$ is defined as in Figure 8. The advantage of F in attacking IBS in an

```
Experiment Expt^{ID-EUF-CMA}_{F(IBS)} (k)
  (mpk, msk) ← MKeyGen(1^k) ; HU ← ∅ ; CU ← ∅
  (S, m, σ) ← F(1^k, mpk; Init(·), Sign(·,·), Corr(·))
  if (S ∈ HU ∧ Verify(mpk, S, m, σ) = 1 ∧ m ∉ M[S])
    then return 1 else return 0
```

```
Oracle Init(id)
  if id ∈ CU ∪ HU
    then return ⊥
  usk[id] ← UKeyGen(msk, id)
  M[id] ← ∅
  HU ← HU ∪ {id}
  return 1
```

```
Oracle Sign(S, m)
  if S ∉ HU
    then return ⊥
  σ ← Sign(usk[S], m)
  M[S] ← M[S] ∪ {m}
  return σ
```

```
Oracle Corr(id)
  if id ∉ HU
    then return ⊥
  CU ← CU ∪ {id}
  HU ← HU \ {id}
  return usk[id]
```

Figure 8: ID-EUF-CMA experiment and oracles

ID-EUF-CMA attack is defined as:

$$\mathrm{Adv}^{\text{ID-EUF-CMA}}_{\mathsf{IBS,F}}(k) \stackrel{\triangle}{=} \Pr\left[\mathsf{Expt}^{\text{ID-EUF-CMA}}_{\mathsf{F(IBS)}}(k) = 1\right] .$$

A UDVS is said to be ID-EUF-CMA-*secure* if no poly-time attacker can get an advantage non-negligible in k, in an ID-EUF-CMA attack against it.

E Example GUDVS Construction

RSA-based UDVS assuming RSA-FDH signature scheme for the signer and registered GQ-type public key for the verifier:

- CPGen simply returns 1^k as the common parameter.
- $\mathsf{VKeyGen}$ is defined as key generation for the GQ protocol, i.e.

$$sk_{\mathrm{v}} = x_{\mathrm{v}} \xleftarrow{\$} \mathbb{Z}^*_{N_{\mathrm{v}}} \text{ and } pk_{\mathrm{v}} = (N_{\mathrm{v}}, e_{\mathrm{v}}, X_{\mathrm{v}}), \text{ where } X_{\mathrm{v}} = x_{\mathrm{v}}^{e_{\mathrm{v}}}.$$

- $\mathsf{SKeyGen}$, Sign and PVer are defined as in RSA-FDH signature, i.e.

$$sk_{\mathrm{s}} = d,\ pk_{\mathrm{s}} = (N_{\mathrm{s}}, e_{\mathrm{s}}), \text{ and } \sigma = H_1(m)^d.$$

- To designate, the signature-holder calculates the DV-signature as follows:

$$\hat{\sigma} \leftarrow \mathsf{SoK}\left\{(\sigma \vee x_{\mathrm{v}}) : \sigma^{e_{\mathrm{s}}} = H_1(m) \mod N_{\mathrm{s}},\ x_{\mathrm{v}}^{e_{\mathrm{v}}} = X_{\mathrm{v}} \mod N_{\mathrm{v}}\right\},$$

  which stands for the following computations:

$$\begin{array}{l} y_{\mathrm{s}} \xleftarrow{\$} \mathbb{Z}^*_{N_{\mathrm{s}}}, Cmt_{\mathrm{s}} \xleftarrow{N_{\mathrm{s}}} y_{\mathrm{s}}^{e_{\mathrm{s}}} \\ Chl_{\mathrm{v}} \xleftarrow{\$} \{0,1\}^{\ell(k)}, Rsp_{\mathrm{v}} \xleftarrow{\$} \mathbb{Z}^*_{N_{\mathrm{v}}}, Cmt_{\mathrm{v}} \xleftarrow{N_{\mathrm{v}}} Rsp_{\mathrm{v}}^{e_{\mathrm{v}}} / X_{\mathrm{v}}^{Chl_{\mathrm{v}}} \\ Chl \leftarrow H_2(pk_{\mathrm{s}}, pk_{\mathrm{v}}, H_1(m), Cmt_{\mathrm{s}}, Cmt_{\mathrm{v}}) \\ Chl_{\mathrm{s}} \xleftarrow{2^{\ell(k)}} Chl - Chl_{\mathrm{v}}, Rsp_{\mathrm{s}} \xleftarrow{N_{\mathrm{s}}} y_{\mathrm{s}} \cdot \sigma^{Chl_{\mathrm{s}}} \\ \hat{\sigma} \leftarrow ((Cmt_{\mathrm{s}}, Cmt_{\mathrm{v}}), (Chl_{\mathrm{s}}, Chl_{\mathrm{v}}, Rsp_{\mathrm{s}}, Rsp_{\mathrm{v}})) \end{array}$$

- To verify the designated signature, one checks if all the following equations hold

$$Cmt_{\mathrm{s}} = \frac{Rsp_{\mathrm{s}}^{e_s}}{H_1(m)^{Chl_{\mathrm{s}}}} \text{ and } Cmt_{\mathrm{v}} = \frac{Rsp_{\mathrm{v}}^{e_{\mathrm{v}}}}{X_{\mathrm{v}}^{Chl_{\mathrm{v}}}} \text{ and } Chl_{\mathrm{s}} + Chl_{\mathrm{v}} = H_2(pk_{\mathrm{s}}, pk_{\mathrm{v}}, H_1(m), Cmt_{\mathrm{s}}, Cmt_{\mathrm{v}}).$$

F More on Proofs of Disjunctive Knowledge

Using the canonical form, the Σ protocol for proof of knowledge of Sec_1 or Sec_2 corresponding to $Pub = (Pub_1, Pub_2)$ can be constructed as in Figure 9, assuming wlog. that the prover knows Sec_1.

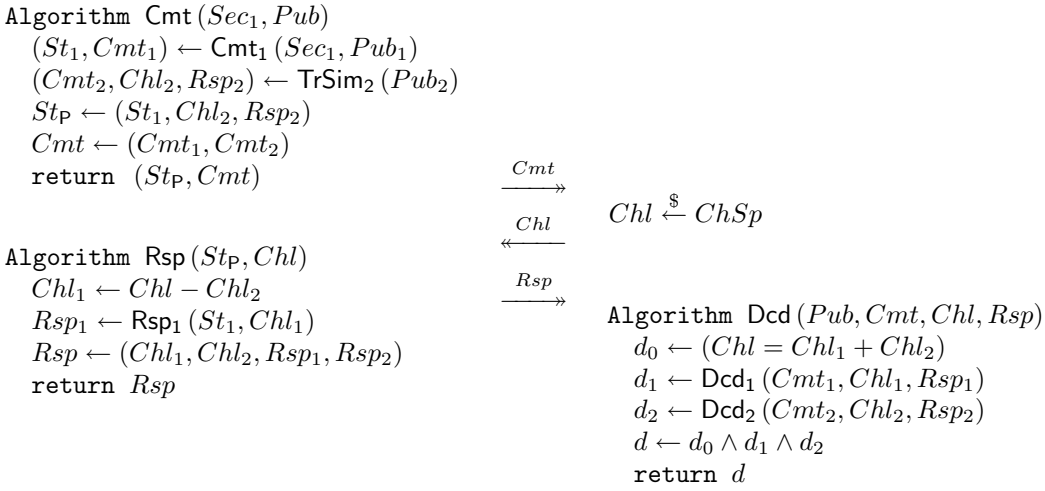

Figure 9: A canonical Σ protocol for proof of disjunctive knowledge

One can easily see that both HVZK and SpS properties are inherited by the constructed proof of disjunctive knowledge. The algorithms for transcript simulation and secret extraction for the protocol in Figure 9 can be constructed as in Figure 10. Again, we assume that Tr and Tr' are such that $Cmt = Cmt'$ (i.e. $(Cmt_1, Cmt_2) =$

```
Algorithm TrSim (Pub)
  (Cmt_1, Chl_1, Rsp_1) ← TrSim_1 (Pub_1)
  (Cmt_2, Chl_2, Rsp_2) ← TrSim_2 (Pub_2)
  Cmt ← (Cmt_1, Cmt_2)
  Chl ← Chl_1 + Chl_2
  Rsp ← (Chl_1, Chl_2, Rsp_1, Rsp_2)
  Tr ← (Cmt, Chl, Rsp)
  return Tr
```

```
Algorithm Ext (Pub, Tr, Tr')
  Tr_1 ← (Cmt_1, Chl_1, Rsp_1)
  Tr'_1 ← (Cmt'_1, Chl'_1, Rsp'_1)
  Tr_2 ← (Cmt_2, Chl_2, Rsp_2)
  Tr'_2 ← (Cmt'_2, Chl'_2, Rsp'_2)
  if Chl_1 ≠ Chl'_1 then
    Sec_1 ← Ext_1 (Pub_1, Tr_1, Tr'_1)
  if Chl_2 ≠ Chl'_2 then
    Sec_2 ← Ext_2 (Pub_2, Tr_2, Tr'_2)
  return (Sec_1, Sec_2)
```

Figure 10: Transcript simulation and extraction algorithms for the construction in Figure 9

$(Cmt'_1, Cmt'_2))$ but $Chl \neq Chl'$. Note that assuming $Chl \neq Chl'$ implies that at least one of the conditions in the extraction algorithm in Figure 10 is correct, thus at least one of the secrets are successfully extracted.

The GQ and Schnorr protocols are respectively for proof of knowledge of RSA roots and discrete logarithms. Following the above convention, we show a GQ protocol for proof of knowledge of the eth RSA root x of X mod N by

$$\mathsf{PoK}\left\{x : x^e = X \mod N\right\},$$

in which $Sec = x$ and $Pub = (N, e, X)$. Furthermore, a Schnorr protocol for proof of knowledge of the discrete logarithm x of X in base g and mod p can be denoted by

$$\mathsf{PoK}\left\{x : g^x = X \mod p\right\},$$

in which $Sec = x$ and $Pub = (p, g, X)$.

Cramer et al's results can be applied to both the GQ and the Schnorr protocols for proving RSA roots and discrete logarithms, respectively. This means that a WI proof of disjunctive knowledge of two RSA roots, i.e.

$$\mathsf{PoK}\left\{(x_1 \vee x_2) :\ x_1^{e_1} = X_1 \mod N_1\ ,\ x_2^{e_2} = X_2 \mod N_2\ \right\},$$

or a WI proof of disjunctive knowledge of two discrete logarithms, i.e.

$$\mathsf{PoK}\left\{(x_1 \vee x_2) :\ g_1^{x_1} = X_1 \mod p_1\ ,\ g_2^{x_2} = X_2 \mod p_2\ \right\},$$

can be constructed.

As also remarked by Cramer et al. [CDS94, as a *remark* on the main theorem], one can observe that their results will still hold even if different protocols are mixed and matched together as long as their respective challenge spaces are the same (and possibly even if they are different). Witness indistinguishability, honest-verifier zero knowledge property, and special soundness property for the resulting construction can be proved using similar techniques to Cramer et al's proofs. Thus, as an example, a WI proof of knowledge of a discrete logarithm or an RSA root, i.e.

$$\mathsf{PoK}\left\{(x_1 \vee x_2) :\ x_1^e = X_1 \mod N\ ,\ g^{x_2} = X_2 \mod p\ \right\}$$

can be constructed as well. Note that both GQ and Schnorr protcols have the same challenge space.

Example Proving knowledge of an x_1 s.t. $x_1^e = X_1 \mod N$ or an x_2 s.t. $g^{x_2} = X_2 \mod p$, i.e.

$$\mathsf{PoK}\left\{(x_1 \vee x_2) :\ x_1^e = X_1 \mod N\ ,\ g^{x_2} = X_2 \mod p\ \right\}$$

can be constructed as follows. The public keys of the two systems are denoted by $pk_1 = (N, e, X_1)$ and $pk_2 = (p, g, X_2)$. There are, of course, two different descriptions of the prover's algorithm, based on whether P knows x_1 or x_2. Let us define $pk = (pk_1, pk_2)$. In the following, we give the two descriptions:

1. description for the case where P knows x_1:

$$
\begin{array}{l}
\texttt{Algorithm } \mathsf{Cmt}\,(x_1, pk) \\
\quad y_1 \xleftarrow{\$} \mathbb{Z}_N^*, Cmt_1 \xleftarrow{N} y_1^e \\
\quad Chl_2 \xleftarrow{\$} \{0,1\}^{\ell(k)}, Rsp_2 \xleftarrow{\$} \mathbb{Z}_p \\
\quad Cmt_2 \xleftarrow{p} g^{Rsp_2}/X_2^{Chl_2} \\
\quad Cmt \leftarrow (Cmt_1, Cmt_2) \\
\quad St_{\mathsf{P}} \leftarrow ((x_1, y_1, N), Chl_2, Rsp_2) \\
\quad \texttt{return } (St_{\mathsf{P}}, Cmt)
\end{array}
\qquad
\begin{array}{c}
\\ \\ \\ \\ \xrightarrow{Cmt} \\ \\ \\ \xleftarrow{Chl} \\ \\ \\ \xrightarrow{Rsp} \\ \\ \\ \\ \\
\end{array}
\qquad
\begin{array}{l}
\texttt{Algorithm } \mathsf{Chl}\,(pk, Cmt) \\
\quad Chl \xleftarrow{\$} \{0,1\}^{\ell(k)} \\
\quad St_{\mathsf{V}} \leftarrow (pk, Cmt, Chl) \\
\quad \texttt{return } (St_{\mathsf{V}}, Chl) \\
\\
\texttt{Algorithm } \mathsf{Dcd}\,(St_{\mathsf{V}}, Rsp) \\
\quad d_0 \leftarrow (Chl \overset{2^{\ell(k)}}{=} Chl_1 + Chl_2) \\
\quad d_1 \leftarrow (Rsp_1^e \overset{N}{=} Cmt_1 \cdot X_1^{Chl_1}) \\
\quad d_2 \leftarrow (g^{Rsp_2} \overset{p}{=} Cmt_2 \cdot X_2^{Chl_2}) \\
\quad d \leftarrow d_0 \wedge d_1 \wedge d_2 \\
\quad \texttt{return } d
\end{array}
$$

$$
\begin{array}{l}
\texttt{Algorithm } \mathsf{Rsp}\,(St_{\mathsf{P}}, Chl) \\
\quad Chl_1 \overset{2^{\ell(k)}}{\leftarrow} Chl - Chl_2 \\
\quad Rsp_1 \xleftarrow{N} y_1 \cdot x_1^{Chl_1} \\
\quad Rsp \leftarrow (Chl_1, Chl_2, Rsp_1, Rsp_2) \\
\quad \texttt{return } Rsp
\end{array}
$$

2. description for the case where P knows x_2:

$$
\begin{array}{l}
\texttt{Algorithm } \mathsf{Cmt}\,(x_2, pk) \\
\quad Chl_1 \xleftarrow{\$} \{0,1\}^{\ell(k)}, Rsp_1 \xleftarrow{\$} \mathbb{Z}_N^* \\
\quad Cmt_1 \xleftarrow{N} Rsp_1^e/X_1^{Chl_1} \\
\quad y_2 \xleftarrow{\$} \mathbb{Z}_p, Cmt_2 \xleftarrow{p} g^{y_2} \\
\quad Cmt \leftarrow (Cmt_1, Cmt_2) \\
\quad St_{\mathsf{P}} \leftarrow (Chl_1, Rsp_1, (x_2, y_2, p)) \\
\quad \texttt{return } (St_{\mathsf{P}}, Cmt)
\end{array}
\qquad
\begin{array}{c}
\\ \\ \\ \\ \xrightarrow{Cmt} \\ \\ \\ \xleftarrow{Chl} \\ \\ \\ \xrightarrow{Rsp} \\ \\ \\ \\ \\
\end{array}
\qquad
\begin{array}{l}
\texttt{Algorithm } \mathsf{Chl}\,(pk, Cmt) \\
\quad Chl \xleftarrow{\$} \{0,1\}^{\ell(k)} \\
\quad St_{\mathsf{V}} \leftarrow (pk, Cmt, Chl) \\
\quad \texttt{return } (St_{\mathsf{V}}, Chl) \\
\\
\texttt{Algorithm } \mathsf{Dcd}\,(St_{\mathsf{V}}, Rsp) \\
\quad d_0 \leftarrow (Chl \overset{2^{\ell(k)}}{=} Chl_1 + Chl_2) \\
\quad d_1 \leftarrow (Rsp_1^e \overset{N}{=} Cmt_1 \cdot X_1^{Chl_1}) \\
\quad d_2 \leftarrow (g^{Rsp_2} \overset{p}{=} Cmt_2 \cdot X_2^{Chl_2}) \\
\quad d \leftarrow d_0 \wedge d_1 \wedge d_2 \\
\quad \texttt{return } d
\end{array}
$$

$$
\begin{array}{l}
\texttt{Algorithm } \mathsf{Rsp}\,(St_{\mathsf{P}}, Chl) \\
\quad Chl_2 \overset{2^{\ell(k)}}{\leftarrow} Chl - Chl_1 \\
\quad Rsp_2 \xleftarrow{p} y_2 + Chl_2 \cdot x_2 \\
\quad Rsp \leftarrow (Chl_1, Chl_2, Rsp_1, Rsp_2) \\
\quad \texttt{return } Rsp
\end{array}
$$

As one can see, the verifiers' sides of the protocols are the same. In fact, from the verifier's perspective, both protocols are the same and he cannot find out if the prover knows x_1 or x_2.

G Bellare and Neven's General Forking Lemma

Lemma 3 *Let $q \geq 1$ and H be a set such that $|H| \geq 2$. Let A be a randomized algorithm that has two outputs, the first of which is an integer in $\{0, 1, \ldots, q\}$. Let also Coins be the set of all possible coins for A. We define the accepting probability of A with respect to an input generator IG as follows*

$$acc \overset{\triangle}{=} \Pr\left[J \geq 1 : x \leftarrow \mathsf{IG}; h_1, \ldots, h_q \xleftarrow{\$} H; (J, \sigma) \leftarrow \mathsf{A}\,(x, h_1, \ldots, h_q)\right] \ .$$

The forker $\mathsf{F_A}$ *is defined as follows*

$$
\begin{array}{l}
\texttt{Algorithm } \mathsf{F_A}\,(x) \\
\quad \rho \xleftarrow{\$} Coins; \quad h_1, \ldots, h_q \xleftarrow{\$} H \\
\quad (J, \sigma) \leftarrow \mathsf{A}\,(x, h_1, \ldots, h_q; \rho) \\
\quad \texttt{if } J = 0 \texttt{ then return } (0, \varepsilon, \varepsilon) \\
\quad h'_J, \ldots, h'_q \xleftarrow{\$} H \\
\quad (J', \sigma') \leftarrow \mathsf{A}\left(x, h_1, \ldots, h_{J-1}, h'_J, \ldots, h'_q; \rho\right) \\
\quad \texttt{if } (J = J' \texttt{ and } h_J \neq h'_J) \texttt{ then return } (1, \sigma, \sigma') \\
\quad\quad \texttt{else return } (0, \varepsilon, \varepsilon)
\end{array}
$$

We also define the success probability of the forker $\mathsf{F_A}$ *with respect to an input generator* IG *as follows*

$$frk \stackrel{\triangle}{=} \Pr\left[b \geq 1 : x \leftarrow \mathsf{IG}; (b, \sigma, \sigma') \leftarrow \mathsf{F_A}\,(x)\right] \quad .$$

Then we have

$$frk \geq acc \cdot \left(\frac{acc}{q} - \frac{1}{|H|}\right) \quad .$$