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

Siamak F Shahandashti[1] and Reihaneh Safavi-Naini[2]

[1] School of Comp. Sci. and Soft. Eng., University of Wollongong, Australia
http://www.uow.edu.au/∼sfs166
[2] Department of Computer Science, University of Calgary, Canada
http://www.cpsc.ucalgary.ca/∼rei

Abstract. We give a generic construction for universal designated-verifier signature schemes from a large class, $\mathbb{C}$, of signature schemes. The resulting schemes are efficient and have two important properties. Firstly, they are provably DV-unforgeable, non-transferable and also non-delegatable. Secondly, the signer and the designated verifier can independently choose their cryptographic settings. We also propose a generic construction for identity-based signature schemes from any signature scheme in $\mathbb{C}$ and prove that the construction is 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 $\mathbb{C}$.

1 Introduction

Universal Designated-Verifier Signatures (UDVS). UDVS schemes were first proposed by Steinfeld et al. [1], based on ideas of Jakobsson et al. [2], with the goal of protecting users' privacy when using certificates. In such a scheme, a user Alice has a certificate that is signed by a certificate issuer. If Alice wants to present her certificate to a verifier Bob, she will use 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, he cannot convince a third party that the certificate was signed by the issuer because he can use his secret key to construct the same designated signature.

Steinfeld et al. proposed security definitions for UDVS schemes and gave a concrete scheme based on bilinear group pairs [1]. In [3] Lipmaa et al. argued that the original security definition in [1] did not sufficiently capture the verifier-designation property and introduced a new security notion, called *non-delegability*. Authors showed that in some UDVS schemes including Steinfeld et al's [1], 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.

As an example, consider the following scenario. A university uses a digital signature scheme to issue student cards. Alice, a student, wants to prove herself a student in a gym to get a discount. To protect her privacy, she converts the university's signature on her card to a designated signature first and then presents the designated signature as a proof of studentship. Now if the UDVS in use is delegatable, the university, without having to issue a card for Alex, a non-student, will be able to publish a value that enables him (and anybody) to compute a designated signature for himself get the discount at the gym. This value does not enable Alex to compute university's private key, sign other documents on behalf of the university, or even compute a designated signature of the university to use other services. Besides, since the university has not actually issued any fraudulent student cards, it cannot be held responsible for any malicious activity. These two facts provide enough safety margin for the university to abuse such delegation ability.

None of the UDVS schemes proposed to date, except a recent scheme of Huang et al. [4], has treated non-delegatability as a security requirement. Furthermore, the results of Lipmaa et al. [3] and later results of Li et al. [5] show that many of the proposed UDVS schemes are delegatable, including the scheme from [1] and one of the schemes from [6].

Our Contributions on UDVS. We give a generic construction for secure UDVS schemes from a large class of signature schemes. The class is defined by requiring certain properties from signature schemes. 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. [3] and adapted to UDVS.

To construct non-delegatable UDVS schemes, we will use Jakobsson et al's approach to providing verifier designation [2]: *"Instead of proving Θ, Alice will prove the statement: Either Θ is true, or I am Bob."* In UDVS schemes, Alice wants to prove validity of her certificate to Bob. A natural construction of UDVS is a 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 as follows: first pick a protocol for proof of knowledge of Alice's certificate and another for the 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. [7]; finally make the resulting protocol non-interactive via e.g. Fiat-Shamir transform [8]. It is intuitively clear that such a construction yields a secure UDVS scheme, assuming both 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 for UDVS schemes that works for any combination of a signature in class $\mathbb{C}$ of signature schemes and all verifier key pairs that belong to a class $\mathbb{K}$, and prove its security in the above sense, in the *Random Oracle Model* (ROM) [9]. The class $\mathbb{C}$ of signatures that can be used in our construction includes signature schemes such as RSA-FDH [10], Schnorr [11], modified ElGamal [12], BLS [13], BB [14], Cramer-Shoup [15], and both schemes proposed by Camenisch and Lysyanskaya [16, 17]. 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 [18], and discrete-log based public and private key pairs.

Our construction are generic and security proofs guarantee security of a large class of UDVS schemes that are obtained from standard signature schemes that are members of the class $\mathbb{C}$. We note that the only other known non-delegatable UDVS due to Huang et al. [4] 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 [19], where he also proposed an *identity-based signature* (IBS) scheme. There are two known generic constructions of IBS. The first is due to Bellare et al. [20], which generalizes an earlier construction of Dodis et al. [21]. They show 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 [22] (called OkDL-IBS by Bellare et al.) and a new IBS scheme proposed in [20] (called BNN-IBS), that are not instances of their generic construction.

The other generic construction is the one of Kurosawa and Heng [23]. 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's secret key is basically a signature of the authority on the user's identity. 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 [23]. Thus, our constructions can be seen as a generalization of theirs.

Many previous IBS schemes can be seen as instances of our generic construction; this includes the schemes of Fiat and Shamir [8], Guillou and Quisquater [18], Shamir [19], pairing-based schemes from [24–29] and basically all the convertible IBS schemes constructed in [20]. 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 ROM. Thus, ROM-free constructions such as the folklore *certificate-based* IBS schemes formalized in [20] and the scheme of Paterson and Schuldt [30] 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 authority 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

UDVS schemes were first proposed by Steinfeld et al. in [1]. The proposed security definitions and a concrete scheme based on bilinear group pairs. In [6] authors proposed extensions of Schnorr and RSA signatures to UDVS schemes. Other pairing-based schemes were proposed in [31] and [32], and Laguillaumie et al. introduced 'Random Oracle free' constructions [33].

Our constructions are very close to Goldwasser and Waisbard's generic constructions of *designated confirmer signatures* in [34]. 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 [35], Gennaro-Halevi-Rabin [36], and Cramer-Shoup [15]. This shows that the above signatures are in class $\mathbb{C}$.

A closely 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 were previously known. Our techniques deal with a similar but different concept of proofs of knowledge of signatures on known messages. Although protocols for proof of knowledge of a secret key corresponding to a public key are more studied and well-known, proof of knowledge of a signature on a message with respect to a known public key has been less studied.

It is worth noting that the previous constructions of identity-based universal deignated verifier signatures by Zhang et al. [37] and universal multi-designated-verifier signatures by Ng et al. [38] are 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 Algorithms, SECURITY NOTIONS, and $\mathcal{O}racles$, respectively. By '$x \leftarrow a$' we denote that a is assigned to x and by '$x \leftarrow \mathsf{X}(a)$' we denote that X with input a is run and the output is assigned to x. $\|$ and $\triangle$ denote concatenation and definition, respectively.

2.2 Proofs of Knowledge

Let P be ab *NP problem* and *Rel* be the corresponding *NP relation*. Let Rel be the corresponding (poly-time) membership deciding algorithm, i.e. $(Pub, Sec) \in Rel$ iff $\mathsf{Rel}(Pub, Sec)$. Following the works of Camenisch and Stadler [39], we will use the notation $\mathsf{PoK}\{Sec : \mathsf{Rel}(Pub, Sec)\}$ for showing a protocol for *proof of knowledge* where the prover proves knowledge of her secret Sec corresponding to a publicly known Pub, s.t. $(Pub, Sec) \in Rel$.

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 a *canonical* form in which the messages sent in the three moves are often called *commitment*, *challenge*, and *response*, denoted here by Cmt, Chl, and Rsp, respectively. The challenge Chl is drawn randomly from a set, called the *challenge space*. The protocol is said to have the *honest-verifier zero-knowledge* property (HVZK) [40], 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 [7], 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. A three-move public-coin protocol with both the HVZK and SpS properties is usually called a Σ protocol.

2.3 Proofs of Disjunctive Knowledge

Cramer et al. showed how to extend Σ protocols to *witness indistinguishable* (WI) Σ protocols for proving knowledge of (at least) t out of n values using secret sharing schemes [7]. They called 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 is known by the prover. 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 [41] 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.

2.4 The Fiat-Shamir Transform

Fiat and Shamir proposed a method for transforming (interactive) three-move public-coin protocols into non-interactive schemes [8]. The idea is to replace the verifier with a hash function. The rationale is that in such a protocols, all the verifier does is providing an unpredictable challenge that can be replaced by a Random Oracle hash function. This idea has been applied in two different ways depending on what is included in the hash function argument. Firstly, the challenge can be set to the hash of the concatenation of the public inputs and the commitment, i.e. $Chl \leftarrow H\,(Pub \parallel Cmt)$. This will result in a *non-interactive proof of knowledge*. We will denote the resulting algorithms for non-interactive proof and verification of knowledge by NIPoK and NIVoK, respectively. Note that the output of the former, denoted by π, is a non-interactive proof that can be publicly verified. HVZK and SpS properties for non-interactive proofs are defined similar to their counterparts in interactive proofs. Pointcheval and Stern's *Forking Lemma* [12] can be used to easily prove in the Random Oracle Model that if the original interactive proof has HVZK and SpS properties then the Fiat-Shamir construction will have these properties too.

A second 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*. Let Sign and Verify denote the resulting algorithms for signing and verification, respectively. Similarly, a signature, denoted by σ, can be verified publicly. The resulting signature scheme will be existentially unforgeable under chosen message attack if the original protocol is a Σ protocol [12, 42, 43].

We use the phrase *signature of knowledge* (SoK) for both the NIPoK and Sign algorithms, and the phrase *verification of knowledge* (VoK) for both the NIVoK and Verify algorithms resulting from applying Fiat-Shamir transform to a Σ protocol as above. Assuming the original protocol is $\mathsf{PoK}\,\{Sec : \mathsf{Rel}\,(Pub, Sec)\}$, we denote the corresponding SoK and VoK by,

$$
\begin{aligned}
&\mathsf{SoK}\,\{Sec : \mathsf{Rel}\,(Pub, Sec)\} \stackrel{\triangle}{=} \mathsf{NIPoK}\,(Pub, Sec)\\
&\mathsf{VoK}\,\{Sec : \mathsf{Rel}\,(Pub, Sec)\}\,(\pi) \stackrel{\triangle}{=} \mathsf{NIVoK}\,(Pub, \pi)\\
&\mathsf{SoK}\,\{Sec : \mathsf{Rel}\,(Pub, Sec)\}\,(m) \stackrel{\triangle}{=} \mathsf{Sign}\,(Pub, Sec, m)\\
&\mathsf{VoK}\,\{Sec : \mathsf{Rel}\,(Pub, Sec)\}\,(m, \sigma) \stackrel{\triangle}{=} \mathsf{Verify}\,(Pub, m, \sigma)\,.
\end{aligned}
$$

2.5 On Public-Private Key Pairs

Key pairs are generated by a *key generation* algorithm KeyGen that takes a security parameter as input and outputs the key pair. In public key systems 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 from a given public key for a key pair, the *underlying problem* of that key pair. A public key thus gives an *instance* of the underlying problem and the corresponding secret key is the *solution* to that problem. 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 *Pair* consisting of all the possible key pairs. 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 $\mathbb{K}$ can be shown as $\mathsf{PoK}\{sk : \mathsf{Pair}(pk, sk)\}$. Some key pairs that have Σ protocols as above are listed in [44]. These include key pairs such as GQ identification scheme, discrete-log-like key pairs, and key pairs of the RSA cryptosystem. We will use the phrase *key type* to refer to the types of the keys. For instance, we denote the keys for the GQ identification scheme by the term 'GQ-type 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 attack (EUF-CMA) [35], is based on the hardness of an *underlying* problem denoted here by P_{SS}. We use $PKSp$ and MSp to denote 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 1. *$\mathbb{C}$ 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* Compute *that on input pk, m, and $\tilde{\sigma}_{\mathrm{aux}}$* 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 $\tilde{\sigma}_{\mathrm{pre}}$ under the one-way function f, i.e. for a converted signature the output of the following algorithm is* `true`*.*

```
Algorithm Valid (pk, m, σ̃)
  (f, I) ← Compute (pk, m, σ̃_aux)
  d ← (f (σ̃_pre) = I)
  return d
```

– *there exists a Σ protocol for proof of knowledge of a $Sec = \tilde{\sigma}_{\mathrm{pre}}$ corresponding to a $Pub = (pk, m, \tilde{\sigma}_{\mathrm{aux}})$ 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*

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

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

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

The properties required by the definition enables the holder of a signature on a message, that is known to a verifier, to efficiently prove the knowledge of the signature, by first converting the signature and then revealing the simulatable part of the converted signature; this will enable 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. Note that since any NP relation has a Σ protocol [45] ensures that for any signature scheme there is a protocol that proves the knowledge of the signature although such protocols are not in general efficient.

Many of the signature schemes in use today fall into the class $\mathbb{C}$. Examples are RSA-FDH [10], Schnorr [11], Modified ElGamal [12], BLS [13], BB [14], Cramer-Shoup [15], Camenisch-Lysyanskaya-02 [16], and Camenisch-Lysyanskaya-04 [17] signatures. In the full version of this paper [44] we briefly show why each of these schemes belongs to $\mathbb{C}$.

4 Universal Designated Verifier Signatures

In this section, we first review the definitions of UDVS schemes and their security. We then propose our generic construction of UDVS schemes from signature schemes in $\mathbb{C}$, and prove its security.

4.1 Definition

A UDVS is a signature scheme with an extra functionality: a holder of a signature can designate the signature to a particular verifier, using the verifier's public key. A UDVS can be described by adding extra algorithms to the ones needed for the the underlying signature scheme. Here, we briefly recall the definitions from Steinfeld et al. [1]. A UDVS has 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; a *Signer (resp. Verifier) Key Generation* algorithms SKeyGen (resp. VKeyGen) that on input 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); a *Signing* and a *Public Verification* algorithm Sign and PVer, where the former takes as input sk_{s} and a message m and outputs a signer's publicly-verifiable (PV) signature σ and the latter takes as input pk_{s} and (m, σ) and outputs a boolean variable for versification result; a *Designation*

and a *Designated Verification* algorithm Desig and DVer, where the former on input pk_s, pk_v, and (m, σ), outputs a designated-verifier (DV) signature $\hat{\sigma}$ and the latter on input pk_s, sk_v, and $(m, \hat{\sigma})$, outputs a boolean verification decision; finally a *Verifier Key-Registration* VKeyReg algorithm, which is a protocol between a *Key Registration Authority* (KRA) and a verifier to register verifier's public key.

4.2 Security

Steinfeld et al. identified two security requirements for UDVS schemes: *DV-unforgeability* and *non-transferability privacy*. We consider a third property proposed by Lipmaa et al. called *non-delegatability*. Intuitively, *DV-unforgeability* captures the inability of the adversary to forge designated signatures on 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 everyone else (everyone except the signature holder and the designated verifier) to generate designated signatures, hence effectively preventing the signer, the signature holder and the designated verifier to delegate their ability to generate designated signatures without revealing their corresponding secrets.

DV-Unforgeability. We use Steinfeld et al's definition of security of UDVS schemes [20] against existential designated signature unforgeability under chosen message attack, denoted by DV-EUF-CMA-attack. For the formal definition refer to [20] or [44].

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 [1]. 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*, such that for any two pairs of keys* (sk_s, pk_s) *and* (sk_v, pk_v) *generated by the key generation algorithms of* UDVS*, and for any message* m*, the following two random variables have the same distribution:*

$$\mathsf{Forge}\,(pk_s, sk_v, pk_v, m) \quad \textit{and} \quad \mathsf{Desig}\,(pk_s, pk_v, m, \mathsf{Sign}\,(sk_s, m)) \quad .$$

Other flavors of non-transferability privacy, i.e. *statistical* and *computational* non-transferability privacy can be analogously defined by requiring the two distributions to be statistically or computationally indistinguishable, respectively.

Non-Delegatability. Lipmaa et al. defined non-delegatability property of designated-verifier signatures [3]. Their definition of κ-non-delegatability requires the designated signature to be a non-interactive *proof of knowledge* with knowledge error κ[46], of the signer's or the designated verifier's secret key. The reason for 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. In a UDVS scheme, we want only the person who holds a signature or the designated verifier be able to produce a designated signature. Lipmaa et al's definition can be 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, with knowledge error κ, of a signature or the designated verifier's secret key.

We use an observation of Cramer et al. [47, p. 359] to simplify the non-delegatability proofs of our constructions. Cramer et al. noted that is that a three-move public-coin protocol with SpS property and challenge space $ChSp$ is a proof of knowledge with knowledge error $\kappa = |ChSp|^{-1}$. Using Forking Lemma, it can be easily seen that the non-interactive version of this observation holds in the Random Oracle Model. 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 have 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 construct a universal designated verifier signature from any signature scheme in $\mathbb{C}$, assuming the verifier has a key pair with key type in $\mathbb{K}$. We use the building blocks introduced before, i.e. proof of disjunctive knowledge and the Fiat-Shamir transform, to construct the UDVS schemes. Our construction has the distinctive property that the verifier's key pair type can be chosen *independently* from the signer's signature. That is the 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 $\mathbb{K}$ with key generation algorithm KeyGen and pair deciding algorithm Pair. The construction is as follows:

- CPGen gets as input 1^k, and returns $cp = 1^k$ as the common parameter. The signer and the verifiers choose their own signature scheme and key pair type,

respectively, i.e.

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

– To designate, the signature-holder first converts the signature and then constructs a signature of disjunctive knowledge of $\tilde{\sigma}_{\text{pre}}$ or sk_{v}. The DV-signature is a pair consisting of $\tilde{\sigma}_{\text{aux}}$ and this signature of knowledge, i.e.

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

– 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 $\tilde{\sigma}_{\text{aux}}$ provided, i.e.

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

4.4 Security Analysis

DV-Unforgeability. We use the *Forking Lemma* to prove DV-Unforgeability of the construction. The Forking Lemma was originally proposed by Pointcheval and Stern [12]. Recently, Bellare and Neven proposed a general version of the Forking Lemma in [48]. We use the results and formulations from the latter in our proof. 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. The proof is given in the full version of this paper [44].

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.*

Non-Transferability Privacy. Non-transferability privacy for GUDVS is due to the very concept behind our construction. The designated signature consists of two values, a publicly-simulatable value $\tilde{\sigma}_{\text{aux}}$ and a *witness indistinguishable* signature of knowledge of a valid converted signature or the verifier's secret key.

Both values are generateable by the designated verifier, indistinguishably from the real designated signatures. To forge a designated signature, the verifier will first simulate $\tilde{\sigma}_{\text{aux}}$ via the algorithm AuxSim and then, similar to the prover, he will be able to construct a non-interactive proof of disjunctive knowledge of $\tilde{\sigma}_{\text{pre}}$ or the verifier's secret key (knowing the latter, of course). The forged designated signature will be consisting of the simulated $\tilde{\sigma}_{\text{aux}}$ along with this signature of knowledge, i.e. we have the following forge algorithm:

$\texttt{Algorithm}\ \mathsf{GUDVS.Forge}\,(pk_{\text{s}}, sk_{\text{v}}, pk_{\text{v}}, m)$
$\quad \tilde{\sigma}_{\text{aux}} \leftarrow \mathsf{AuxSim}\,(pk_{\text{s}}, m)$
$\quad \delta \leftarrow \mathsf{SoK}\,\{(\tilde{\sigma}_{\text{pre}} \vee sk_{\text{v}}) : \mathsf{Valid}\,(pk_{\text{s}}, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}))\,, \mathsf{Pair}\,(pk_{\text{v}}, sk_{\text{v}})\}$
$\quad \hat{\sigma} \leftarrow (\tilde{\sigma}_{\text{aux}}, \delta)$
$\quad \texttt{return}\ \hat{\sigma}$

AuxSim's ability to simulate $\tilde{\sigma}_{\text{aux}}$ and witness indistinguishability of the signature of knowledge together, will imply that the output of the algorithm GUDVS.Forge is indistinguishable from real designated signatures. The existence of AuxSim and a Σ protocol for the 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's 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. The very design of our UDVS construction is geared towards providing 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.

It can be seen 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. This 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 may *not* be 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.

The construction can also be extended to designate more than one signature at a time. This is useful 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 certificates 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, after seeing a designated signature, can not determine which k certificates is used by the user.

4.6 Comparison

We use constructions in [1, 6] as benchmarks for our constructions. We choose instances of our constructions that match the signature scheme and verifier key type of the benchmark schemes. Similar to [6], we assume the cost of computing a product $a^x \cdot b^y \cdot c^z$ and $O(\alpha)$ low exponent exponentiations both, are equivalent to a single exponentiation. We use the same typical parameters for lengths of members of different groups, namely 1.024 kb for DL groups and RSA modules and 0.16 kb for $ChSp$. To further simplify the comparison, 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 note that designation of a certificate has two phases: before choosing the designated verifier and after that and so computation can be carried out in accordingly. We *off-line* and *on-line* to denote the two phases, respectively. An interesting property of our construction is that cost of on-line phase is relatively low (one multiplication). This makes our constructions suitable for systems in which certificates must be frequently verified by (and hence designated to) multiple different verifiers. Table 1 summarizes the

comparisons, with "Typ. $\hat{\sigma}$ len." and "ND" standing for "Typical $\hat{\sigma}$ length" and "Non-Delegatability", respectively and comparatively more desirable values in bold. 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 online designation cost and provide provable non-delegatabilty. Our schemes are also (almost) generic and provide the desirable property of *signer-verifier setting independence*. A side effect of using the Forking Lemma for proof of security is that security reductions are not *tight*.

Table 1. Comparison of the properties of Steinfeld et al's schemes with those of their corresponding GUDVS counterparts

Scheme	Hard problem	Desig cost		DVer cost	Typ. $\hat{\sigma}$ len.	ND
		off-line	on-line			
DVSBM [1]	BDH	**none**	1 pair.	**1 pair.**	**1.0 kb**	✗
GUDVS (BLS+DL)	**CDH**	2 pair.	**1 mult.**	2 pair.	5.3 kb	✓
SchUDVS$_1$ [6]	SDH	**1 exp.**	1 exp.	**1 exp.**	2.0 kb	✗
SchUDVS$_2$ [6]	**DL**	2 exp.	1 exp.	2 exp.	**1.5 kb**	?
GUDVS (Schnorr+DL)	**DL**	4 exp.	**1 mult.**	3 exp.	5.3 kb	✓
RSAUDVS [6]	**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**	✓

5 Identity-based Signatures

In this section, we first review the definitions of the IBS scheme and its security. Then we propose a 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 [19] in an attempt to remove the need for a public-key infrastructure. In such systems, the users' identities are used as their public keys. However, users lose their ability to choose their own secret keys and must ask 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 1^k, where k is a security parameter, outputs a pair of master secret key and master public key (msk, mpk), a *user key generation* algorithm UKeyGen, which on input msk and a user identity id, outputs a user secret key usk, a *signing* algorithm Sign, which on input usk and a message m, outputs a signature σ on the message, and finally a *verification* algorithm Verify, which on input mpk, id, and (m, σ), outputs a binary

decision indicating whether or not σ is a valid signature on m with respect to mpk and id.

We use Bellare and Neven's definition for the security of an IBS scheme [20] against existential unforgeability under chosen message and identity attacks, denoted by ID-EUF-CMA-attack. For the formal definition refer to [20] or [44].

5.2 Generic Construction of IBS and Its Security

In this section we show how to extend a signature in $\mathbb{C}$ to an IBS scheme. The idea is to use the key pair of the signature scheme as the master key pair of KGC, and use the signing algorithm as the users' key generation algorithm in the following way: a user's secret key corresponding to her public identity, is obtained by signing the user's identity using the KGC's secret key. The secret key is securely given to the user. Now, the user is able to prove her identity, since she can prove the 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. Let that the standard signature $\mathsf{SS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be 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 public and secret key pair as the master public and secret key pair for the identity based signature scheme. To generate a user's key pair, the KCG simply signs the user's identity using his master secret key and outputs the generated signature (together with the master public key and the user's identity) as the user's secret key, i.e.

$$
\begin{array}{l}
\texttt{Algorithm } \mathsf{GIBS.MKeyGen}\,(k) \\
\quad (msk, mpk) \leftarrow \mathsf{SS.KeyGen}\,(k) \\
\quad \texttt{return } (msk, mpk)
\end{array}
\qquad
\begin{array}{l}
\texttt{Algorithm } \mathsf{GIBS.UKeyGen}\,(msk, id) \\
\quad \sigma \leftarrow \mathsf{SS.Sign}\,(msk, id) \\
\quad usk \leftarrow (mpk, id, \sigma) \\
\quad \texttt{return } usk
\end{array}
$$

An identity-based signature is constructed as a signature of knowledge of KGC's signature on the identity of the signer by, first running the 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 constructing a proof of knowledge of $\tilde{\sigma}_{\text{pre}}$ and, finally transforming the result 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, i.e.

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

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.

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

This construction is a generalized version of Kurosawa and Heng's construction [23]. They required a stronger requirement on their signature schemes. We note the similarities between the ideas behind Kurosawa and Heng's and our constructions, and that of Naor's observation on transforming any identity-based encryption to a standard signature scheme [49, p. 226]. In both, a user's secret key is a signature of the KGC on the user's identity. Our constructions can be seen as the Naor's observation in the reverse direction, 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 the full version of this paper [44].

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 generic construction of IBS schemes has kind of a *nesting* property in the sense 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 to 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 above, 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 [50].

An extension of the GIBS construction that follows from the nesting property is the construction of *identity-based universal designated verifier signatures* from any signature in $\mathbb{C}$. In such a scheme, a designator wishes to designate a certificate signed by an identity-based signature, 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 such a 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 proposed generic constructions of UDVS and IBS schemes from a large class of signatures. Our constructions result in schemes which have comparable efficiency to those with similar properties. The generic UDVS construction is provably non-delegatable and offers a desirable property, which is independence of the signer's and the verifier's setting. 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

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 this work.

References

1. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal Designated-Verifier Signatures. In Laih, C.S., ed.: ASIACRYPT. Volume 2894 of Lecture Notes in Computer Science., Springer (2003) 523–542
2. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: EUROCRYPT. (1996) 143–154
3. Lipmaa, H., Wang, G., Bao, F.: Designated Verifier Signature Schemes: Attacks, New Security Notions and a New Construction. In Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M., eds.: ICALP. Volume 3580 of Lecture Notes in Computer Science., Springer (2005) 459–471
4. Huang, X., Susilo, W., Mu, Y., Wu, W.: Universal Designated Verifier Signature Without Delegatability. In Ning, P., Qing, S., Li, N., eds.: ICICS. Volume 4307 of Lecture Notes in Computer Science., Springer (2006) 479–498
5. Li, Y., Lipmaa, H., Pei, D.: On Delegatability of Four Designated Verifier Signatures. In Qing, S., Mao, W., Lopez, J., Wang, G., eds.: ICICS. Volume 3783 of Lecture Notes in Computer Science., Springer (2005) 61–71

6. Steinfeld, R., Wang, H., Pieprzyk, J.: Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures. [51] 86–100
7. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Desmedt, Y., ed.: CRYPTO. Volume 839 of Lecture Notes in Computer Science., Springer (1994) 174–187
8. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Odlyzko, A.M., ed.: CRYPTO. Volume 263 of Lecture Notes in Computer Science., Springer (1986) 186–194
9. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM Conference on Computer and Communications Security. (1993) 62–73
10. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: EUROCRYPT. (1996) 399–416
11. Schnorr, C.P.: Efficient Signature Generation by Smart Cards. J. Cryptology **4** (1991) 161–174
12. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. J. Cryptology **13** (2000) 361–396
13. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In Boyd, C., ed.: ASIACRYPT. Volume 2248 of Lecture Notes in Computer Science., Springer (2001) 514–532
14. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. [52] 56–73
15. Cramer, R., Shoup, V.: Signature Schemes Based on the Strong RSA Assumption. ACM Trans. Inf. Syst. Secur. **3** (2000) 161–185
16. Camenisch, J., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In Cimato, S., Galdi, C., Persiano, G., eds.: SCN. Volume 2576 of Lecture Notes in Computer Science., Springer (2002) 268–289
17. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In Franklin, M.K., ed.: CRYPTO. Volume 3152 of Lecture Notes in Computer Science., Springer (2004) 56–72
18. Guillou, L.C., Quisquater, J.J.: A "Paradoxical" Indentity-Based Signature Scheme Resulting from Zero-Knowledge. In Goldwasser, S., ed.: CRYPTO. Volume 403 of Lecture Notes in Computer Science., Springer (1988) 216–231
19. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: CRYPTO. (1984) 47–53
20. Bellare, M., Namprempre, C., Neven, G.: Security Proofs for Identity-Based Identification and Signature Schemes. [52] 268–286
21. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong Key-Insulated Signature Schemes. [53] 130–144
22. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. [54] 31–53
23. Kurosawa, K., Heng, S.H.: From Digital Signature to ID-based Identification/Signature. [51] 248–261
24. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. Sympoium on Cryptography and Information Security (SCIS), Okinawa, Japan (2000) 26–28
25. Hess, F.: Efficient Identity Based Signature Schemes Based on Pairings. In Nyberg, K., Heys, H.M., eds.: Selected Areas in Cryptography. Volume 2595 of Lecture Notes in Computer Science., Springer (2002) 310–324
26. Cha, J.C., Cheon, J.H.: An Identity-Based Signature from Gap Diffie-Hellman Groups. [53] 18–30
27. Yi, X.: An Identity-Based Signature Scheme from the Weil Pairing. Communications Letters, IEEE **7** (2003) 76–78

28. Barreto, P.S.L.M., Libert, B., McCullagh, N., Quisquater, J.J.: Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In Roy, B.K., ed.: ASIACRYPT. Volume 3788 of Lecture Notes in Computer Science., Springer (2005) 515–532
29. Huang, Z., Chen, K., Wang, Y.: Efficient Identity-Based Signatures and Blind Signatures. In Desmedt, Y., Wang, H., Mu, Y., Li, Y., eds.: CANS. Volume 3810 of Lecture Notes in Computer Science., Springer (2005) 120–133
30. Paterson, K.G., Schuldt, J.C.N.: Efficient Identity-Based Signatures Secure in the Standard Model. In Batten, L.M., Safavi-Naini, R., eds.: ACISP. Volume 4058 of Lecture Notes in Computer Science., Springer (2006) 207–222
31. Zhang, R., Furukawa, J., Imai, H.: Short Signature and Universal Designated Verifier Signature Without Random Oracles. In Ioannidis, J., Keromytis, A.D., Yung, M., eds.: ACNS. Volume 3531 of Lecture Notes in Computer Science. (2005) 483–498
32. Vergnaud, D.: New Extensions of Pairing-Based Signatures into Universal Designated Verifier Signatures. In Bugliesi, M., Preneel, B., Sassone, V., Wegener, I., eds.: ICALP (2). Volume 4052 of Lecture Notes in Computer Science., Springer (2006) 58–69
33. Laguillaumie, F., Libert, B., Quisquater, J.J.: Universal Designated Verifier Signatures Without Random Oracles or Non-black Box Assumptions. In Prisco, R.D., Yung, M., eds.: SCN. Volume 4116 of Lecture Notes in Computer Science., Springer (2006) 63–77
34. Goldwasser, S., Waisbard, E.: Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes. In Naor, M., ed.: TCC. Volume 2951 of Lecture Notes in Computer Science., Springer (2004) 77–100
35. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. **17** (1988) 281–308
36. Gennaro, R., Halevi, S., Rabin, T.: Secure Hash-and-Sign Signatures Without the Random Oracle. In: EUROCRYPT. (1999) 123–139
37. Zhang, F., Susilo, W., Mu, Y., Chen, X.: Identity-Based Universal Designated Verifier Signatures. In Enokido, T., Yan, L., Xiao, B., Kim, D., Dai, Y., Yang, L.T., eds.: EUC Workshops. Volume 3823 of Lecture Notes in Computer Science., Springer (2005) 825–834
38. Ng, C.Y., Susilo, W., Mu, Y.: Universal Designated Multi Verifier Signature Schemes. In: ICPADS (2), IEEE Computer Society (2005) 305–309
39. Camenisch, J., Stadler, M.: Efficient Group Signature Schemes for Large Groups (Extended Abstract). In Jr., B.S.K., ed.: CRYPTO. Volume 1294 of Lecture Notes in Computer Science., Springer (1997) 410–424
40. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Comput. **18** (1989) 186–208
41. Camenisch, J., Stadler, M.: Proof Systems For General Statements about Discrete Logarithms. Technical Report 260, Dept. of Computer Science, ETH Zurich (1997)
42. Ohta, K., Okamoto, T.: On Concrete Security Treatment of Signatures Derived from Identification. In Krawczyk, H., ed.: CRYPTO. Volume 1462 of Lecture Notes in Computer Science., Springer (1998) 354–369
43. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security. In Knudsen, L.R., ed.: EUROCRYPT. Volume 2332 of Lecture Notes in Computer Science., Springer (2002) 418–433

44. Shahandashti, S.F., Safavi-Naini, R.: Construction of Universal Designated-Verifier Signatures and Identity-Based Signatures from Standard Signatures. Cryptology ePrint Archive, Report 2007/462 (2007) http://eprint.iacr.org/.
45. Catalano, D., Dodis, Y., Visconti, I.: Mercurial Commitments: Minimal Assumptions and Efficient Constructions. In Halevi, S., Rabin, T., eds.: TCC. Volume 3876 of Lecture Notes in Computer Science., Springer (2006) 120–144
46. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. [54] 390–420
47. Cramer, R., Damgård, I., MacKenzie, P.D.: Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In Imai, H., Zheng, Y., eds.: Public Key Cryptography. Volume 1751 of Lecture Notes in Computer Science., Springer (2000) 354–372
48. Bellare, M., Neven, G.: Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. In Juels, A., Wright, R.N., di Vimercati, S.D.C., eds.: ACM Conference on Computer and Communications Security, ACM (2006) 390–399
49. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. In Kilian, J., ed.: CRYPTO. Volume 2139 of Lecture Notes in Computer Science., Springer (2001) 213–229
50. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In Zheng, Y., ed.: ASIACRYPT. Volume 2501 of Lecture Notes in Computer Science., Springer (2002) 548–566
51. Bao, F., Deng, R.H., Zhou, J., eds.: Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004. In Bao, F., Deng, R.H., Zhou, J., eds.: Public Key Cryptography. Volume 2947 of Lecture Notes in Computer Science., Springer (2004)
52. Cachin, C., Camenisch, J., eds.: Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. In Cachin, C., Camenisch, J., eds.: EUROCRYPT. Volume 3027 of Lecture Notes in Computer Science., Springer (2004)
53. Desmedt, Y., ed.: Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings. In Desmedt, Y., ed.: Public Key Cryptography. Volume 2567 of Lecture Notes in Computer Science., Springer (2002)
54. Brickell, E.F., ed.: Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings. In Brickell, E.F., ed.: CRYPTO. Volume 740 of Lecture Notes in Computer Science., Springer (1993)