Non-interactive Confirmer Signatures

Sherman S.M. Chow[1,⋆] and Kristiyan Haralambiev[2]

[1] Department of Combinatorics and Optimization
University of Waterloo, Ontario, Canada N2L 3G1
smchow@math.uwaterloo.ca
[2] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
kkh@cs.nyu.edu

Abstract. The study of non-transferability of digital signatures, such as confirmer signatures, has enjoyed much interest over the last twenty years. In PKC '08, Liskov and Micali noted that all previous constructions of confirmer signatures consider only offline untransferability – non-transferability is not preserved if the recipient interacts concurrently with the signer/confirmer and an unexpected verifier. We view this as a result of all these schemes being interactive in the confirmation step. In this paper, we introduce the concept of non-interactive confirmer signatures (which can also be interpreted as extractable universal designated-verifier signatures). Non-interactive confirmer signatures give a neat way to ensure the online untransferability of signatures. We realize our notion under the "encryption of a signature" paradigm using pairings and provide a security proof for our construction without random oracles.

Keywords: non-interactive confirmer signature, extractable universal designated verifier signature, online untransferability.

1 Introduction

Non-transferability of digital signatures is an interesting research problem that has been investigated in various works over the last twenty years. A canonical application considers the scenario in which Alice wants to make an offer to Bob, but does not want Bob to show it to anybody else, so that Bob can not use Alice's offer as leverage to negotiate better terms or to gain any advantage. This covers the scenarios of job offers, contracts, receipt-free elections, and selling of malware-free software.

1.1 Undeniable Signatures and Confirmer Signatures

To address this problem, Chaum and van Antwerpen [1] introduced the notion of *undeniable signatures* which requires the signer's presence and cooperation for the recipient to verify the validity of a signature. In this way, the signer

⋆ A major part of the work was done while at New York University.

A. Kiayias (Ed.): CT-RSA 2011, LNCS 6558, pp. 49–64, 2011.

controls when the validity of the signature is being confirmed, and the validity is unknown without the participation of the signer. With this extra power of the signer, a basic security requirement is that the signer cannot cheat about the (in)validity of an undeniable signature when participating in the confirmation/disavowal protocols. However, there is no cryptographic means which can prevent a signer from refusing to cooperate. If the signer becomes unavailable or decides to "repudiate" the signature by ignoring any confirmation requests, the recipient is left with no cryptographic evidence of the signature's validity.

To overcome this disadvantage and to better ensure *non-repudiation*, Chaum [2] introduced a confirmer in this setting, which is a party other than the signer who can confirm/deny a signature. Now the trust (of willingness to participate in the protocol) is moved from the signer to the confirmer. Furthermore, this confirmer can extract an ordinary digital signature that is publicly and non-interactively verifiable (say when Bob has accepted the offer but Alice denies making one). This notion is known as designated confirmer signatures. In this paper, we follow the naming of some recent work and call it confirmer signatures.

Recently, Liskov and Micali [3] pointed out that all constructions of confirmer signatures provide only *offline untransferability*, and possibly Bob can "transfer" the validity of the signature by interacting with Alice and a verifier concurrently. They propose the notion of online-untransferable signatures to address this problem. However, their construction is inefficient due to the use of "cut-and-choose" proofs, i.e., the number of cryptographic operations like encryption and signing is linear in the security parameter. Also, the confirmer needs to either setup a public key for *each* signer, or to use an identity-based encryption (IBE) for the extraction of a publicly-verifiable signature. Both could be viewed as shortcomings of their construction, or the complexity one needs to pay to achieve online untransferability in an interactive setting. Lastly, their definitions deviate from the standard ones due to the absence of the confirmation protocol, which might be needed if the confirmer has to convince a verifier different from the recipient of a signature (e.g., in cases of checking integrity-critical content as part of a subscription service [4]). It is fair to say previous constructions either are inefficient or provide only offline untransferability.

1.2 (Universal) Designated-Verifier Signatures

Shortly after Chaum's work, Jakobsson *et. al.* [5] observed that undeniable signatures allow the signer to choose only whether to engage in the confirm/disavowal protocol but not with whom, i.e., it is possible that the recipient acts as a man-in-the-middle and executes the confirmation protocol with the signer so as to convince a third party. Moreover, this puts the signer at risk of being coerced and forced to participate in the confirmation protocol. To address these problems, they suggested the idea of *designated verifier proofs* which allows the prover (signer) to designate who will be convinced by the proof. If Alice wanted to convince Bob of the validity of a signature, or generally a statement θ, then Alice would prove to Bob that "*either* θ *is true, or I am Bob*". That would definitely

convince Bob, but if he tried to transfer the proof to others, it would not imply anything about the validity of θ simply because the proof came from Bob.

Steinfeld *et. al.* [6] generalized this notion to *universal designated-verifier signatures* (UDVS). Unlike the original definition, one does not have to be the signer in order to convince a designated verifier. Anyone who is in possession of a regular signature can perform such a proof to a designated verifier $\mathcal{V}$. Their construction requires the verifiers to register their public/private key pairs with a key registration authority using the same system parameters as the signature scheme of the signer. This key registration model is a fairly common requirement when using public key infrastructure.

The original application of UDVS is motivated by privacy concerns associated with dissemination of signed digital certificates. An universal designator, which means any receiver of a certificate, can transfer the validity of the signature to any designated verifier. There is no mean to extract an ordinary (publicly-verifiable) signature from a UDVS. One way to do this is to ask the universal designator to hold the original signature. Alice still needs to trust that the signature will be kept confidential by this designator, i.e., active collusion with Bob or attack by Bob would not happen. Bob must ask Alice or the confirmer for the signature, which means Bob need to place trust on this middle-man too. In other words, we still need to make trust assumptions which may be implicitly made when using confirmer signatures. Simply put, the UDVS may provide online-untransferability, but not non-repudiation, as also pointed out in [3].

We remark that the universal designated verifier signature proof proposed by Baek *et. al.* [7] (which is later renamed to credential ownership proof [8]) does not has any requirement about whether the verifier cannot convince a third party that the message has been actually signed by a signer. In their proof systems, interactive protocols between the signature holder and the designated verifier are required to avoid the key requirement for the verifiers.

1.3 Related Work

Boyar *et. al.* [9] introduced the concept of convertible undeniable signatures, which allows the possibility of converting either a single undeniable signature or all undeniable signatures ever produced by a signer into an ordinary one. Similar to traditional undeniable/confirmer signature, confirmation is interactive.

ElAimani revisited in a series of work [10,11] the construction of undeniable or confirmer signature following the "encryption of a signature" paradigm firstly studied by Okamoto [12]. These studies identified the minimal security required for the encryption scheme in the generic constructions of strongly-unforgeable schemes which can be instantiated by a rather wide class of signature schemes. Since our scheme is also constructed using the "encryption of a signature" approach, all these constructions share similarity (except some technical details such as an ordinary signature[1] can be extracted from our scheme since we do

[1] By an ordinary signature, we mean that the signature just signs on the message of interest but includes nothing else. For example, there should be no other auxiliary information that is only useful for (the security of) the confirmer signature.

not aim at getting strong unforgebility). We view the merit of our work as a variation of the traditional paradigm which gives a conceptually simple solution to the online transferability problem.

1.4 Our Contribution

The research focus of confirmer signatures has been on defining the right model and constructing efficient schemes. We follow these directions here by introducing the concept of *non-interactive* confirmer signatures (NICS) and the first efficient (non-interactive) confirmer signature scheme with *online untransferability*. Our construction is secure in the key registration model without random oracles. This result can also be interpreted as *extractable* universal designated-verifier signatures (xUDVS), i.e., one could extract the underlying regular signature if in possession of the private extraction key. This extraction key pair is a single key pair for normal public key encryption, but not an extra key pair generated by a confirmer for each signer, nor a master public/private key pair for an identity-based encryption (c.f. [3]). Surprisingly, the works studying confirmer signatures and designated verifier proofs/signatures have been almost entirely independent despite both are originating from the same problem.

The correspondences between an NICS and an xUDVS are as follow. Signing and extraction are the same. Confirmation is done by taking an ordinary signature as an input and creating a new NICS/xUDVS with respect to *any* verifier who asked for a confirmation (in contrast to [3]). This process can be possibly done by the signer herself, or any holder of an ordinary signature (i.e., a universal confirmation) At the same time, it is possible to extract an ordinary signature out of the NICS/xUDVS by using the private key of a third-party.

The benefits of non-interactive confirmation are numerous. It provides non-transferability of signatures in a natural way and simplifies the traditional way of "sending an unverifiable signature to the verifier first, then the verifier asks for the signer/confirmer to confirm/disavow later". In particular, the disavowal protocol is not necessary any more. Most importantly, a non-interactive confirmer signature scheme avoids the problem of the recipient interacting concurrently with the signer and another verifier. So, the online untransferability is easily satisfied. We will see shortly that the security definition also becomes neater.

2 Non-interactive Model for Confirmer Signatures

2.1 Notations

Let $\mathsf{negl}(\kappa)$ denote a negligible function in κ where a function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ is said to be negligible if for all $c > 0$ and sufficiently large κ, $\epsilon(\kappa) \leq \kappa^{-c}$. For a finite set S, we denote $x \in_R S$ the sampling of an element x from S uniformly at random. If $\mathcal{A}$ is a PPT algorithm, $\mathcal{A}(x)$ denotes the output distribution of $\mathcal{A}$ on input x. We write $y \leftarrow \mathcal{A}(x)$ to denote the experiment of running $\mathcal{A}$ on input x and assigning the output to the variable y. Also, let

$$\Pr[x_1 \leftarrow X_1;\ x_2 \leftarrow X_2(x_1);\ \ldots;\ x_n \leftarrow X_n(x_1, \ldots, x_{n-1}) : \rho(x_1, \ldots, x_n)]$$

be the probability that the predicate $\rho(x_1, \ldots, x_n)$ is true when x_1 is sampled from the distribution X_1; x_2 is sampled from the distribution $X_2(x_1)$ which possibly depends on x_1; $x_3, \ldots, x_{n-1}$ are defined similarly; and finally x_n is sampled from distribution $X_n(x_1, \ldots, x_{n-1})$ which possibly depends on $x_1, \ldots, x_{n-1}$. The predicate might include execution of probabilistic algorithms.

2.2 Framework

Before describing different algorithms required in a non-interactive confirmer signature scheme, we first introduce the four kinds of participants involved, namely, signers, verifiers, adjudicator, and universal confirmer. The role of the former two are obvious. An adjudicator is mostly a passive entity who is assigned by a signer (possibly with the consent of the verifier given outside of our protocols) in the creation of a confirmer signature. A verifier may turn to an adjudicator when the signer refused to give an ordinary signature afterwards.

The role of a universal confirmer is not the same as a traditional one. For traditional confirmer/undeniable signatures, the signatures generated by the signer is ambiguous, i.e., it does not bind to the signer by itself. The job of a confirmer is to convince the verifier about the validity of a signature, which also means that the confirmer must maintain some secret information other than the signature itself, may it be a private key or some random values used by the signer during signature generation; otherwise anyone can confirm the validity of a signature.

In our setting, the signing process started by generating a regular signature that is binding to the signer. Consequently, there must be a step which converts an ordinary signature to an ambiguous one. We include this step as part of the confirmation. While the signature produced is ambiguous, the confirmation can still convince the verifier that the signer has signed on a particular message. Moreover, this can also confirm the fact that an ordinary signature by the signer on this message can also be extracted by an adjudicator. The confirmer in our notion is universal, which means that anyone who holds an ordinary signature can do this job. The confirmer does not need to maintain additional secret state information. Of course, the signer may perform the confirmation herself as well.

Now we are ready to define a non-interactive confirmer signature scheme with a global setup. Specifically, this setup decides the security parameter κ, the cryptographic groups to be used, and possibly a common reference string. All these will be included in the system parameters param, implicitly required by all algorithms. Like existing constructions, we require both signers and verifiers to register a public (verification) key with the key registration authority. One way to do that is to prove knowledge of the secret key during key registration.

Definition 1 (Non-Interactive Confirmer Signatures). *A non-interactive confirmer signature scheme is a signature scheme* $(\mathsf{SKGen}, \mathsf{Sig}, \mathsf{Ver})$ *augmented with two suites of algorithms: First, for convincing a designated verifier, we have:*

- $\mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \sigma, m)$*: takes as inputs an unordered pair of verification keys (one of the signer and one of the verifier), an adjudicator public key* pk*, and a signature/message pair which is valid for any key among* $\{\mathsf{vk}_S, \mathsf{vk}_V\}$*; outputs a confirmer signature* $\hat{\sigma}$*.*

– $\mathsf{DVer}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \hat{\sigma}, m)$*: takes as inputs an unordered pair of verification keys, a confirmer signature* $\hat{\sigma}$ *and a message* m*; outputs* 1 *if* $\hat{\sigma}$ *is an output of* $\mathsf{Des}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \sigma, m)$*, where* σ *is a signature of* m *verifiable under* vk_0 *or* vk_1*; otherwise, outputs 0.*

For extracting an ordinary signature for a verifier, we have:

– $\mathsf{EKGen}(1^\kappa)$*: outputs a private/public key* $(\mathsf{xk}, \mathsf{pk})$ *for the adjudicator.*
– $\mathsf{Ext}(\mathsf{xk}, \hat{\sigma})$*: takes as inputs the private extraction key and a valid (publicly verifiable) confirmer signature outputted by* $\mathsf{Des}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \sigma, m)$*, outputs an ordinary signature* σ *and a bit* b *indicating that* σ *is given under the signing key corresponding to* vk_b*.*

In the traditional notion of confirmer signatures, the job of our adjudicator about extracting an ordinary signature is also performed by the confirmer. Here we distill this task out. As argued before, a confirmer may be absent, as a designated verifier can "confirm" the signature $\hat{\sigma}$ by himself. On the other hand, any holder of an ordinary signature σ can designate the proof to any verifier, similar to the functionality of the universal designated verifier signatures.

2.3 Security Requirements

Our model borrows ideas from both confirmer signatures and universal designated verifier signatures. Compared with traditional confirmer signatures, our definition is considerably neater since the confirmation and disavowal protocols are essentially replaced by a single Des algorithm. Most importantly, all previous constructions are interactive and their security requirements are defined with respect to that. We see that as a reason why all these works can only achieve offline untransferability and a more complex definition is required to ensure online untransferability. We believe that our definition satisfy all fundamental properties of confirmer signatures. The following requirements should be satisfied for all system parameters param generated by the global setup.

Definition 2 (Correctness). *A non-interactive confirmer signature scheme (of security parameter* κ*) is correct if the four conditions below are satisfied (with overwhelming probability in* κ*).*

– ***Key-Correctness***
We assume it is efficient to check for the validity of all kinds of key pairs in our system. We denote this check by the predicate $\mathsf{Valid}(\cdot)$ *which takes as implicit input the key pair type. Specifically, we require that* $\mathsf{Valid}(\mathsf{sk}, \mathsf{vk}) = 1$ *and* $\mathsf{Valid}(\mathsf{xk}, \mathsf{pk}) = 1$ *for all* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}(1^\kappa)$ *and* $(\mathsf{xk}, \mathsf{pk}) \leftarrow \mathsf{EKGen}(1^\kappa)$*.*
– ***Sign-Correctness***
For all messages m*, and all* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}(1^\kappa)$*,* $\mathsf{Ver}(\mathsf{vk}, m, \mathsf{Sig}(\mathsf{sk}, m)) = 1$*.*
– ***Designate-Correctness***
For all messages m*, all* $(\mathsf{sk}_S, \mathsf{vk}_S), (\mathsf{sk}_V, \mathsf{vk}_V) \leftarrow \mathsf{SKGen}(1^\kappa)$*, and all* $(\mathsf{xk}, \mathsf{pk}) \leftarrow \mathsf{EKGen}(1^\kappa)$*, and all* $\hat{\sigma} \leftarrow \mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \mathsf{Sig}(\mathsf{sk}_S, m), m)$*, we expect that* $\mathsf{DVer}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \hat{\sigma}, m) = 1$*, where* $\{\mathsf{vk}_S, \mathsf{vk}_V\}$ *is ordered lexicographically.*

– ***Extract-Correctness***
For all messages m*, all* $(\mathsf{sk}_0, \mathsf{vk}_0), (\mathsf{sk}_1, \mathsf{vk}_1) \leftarrow \mathsf{SKGen}(1^\kappa)$*, all* $(\mathsf{xk}, \mathsf{pk}) \leftarrow \mathsf{EKGen}(1^\kappa)$*, all* $b \in \{0,1\}$*, all* $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_b, m)$*, and finally for all* $\hat{\sigma} \leftarrow \mathsf{Des}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \sigma, m)$*, it is true that* $\mathsf{Ext}(\mathsf{xk}, \hat{\sigma}) = (\sigma, b)$.

Privacy. Regarding the privacy requirement of the confirmer signatures, some existing works consider invisibility, which informally means that an adversary $\mathcal{A}$ with the signing keys and accesses to extraction oracles cannot distinguish whether a confirmer signature is on which of the two chosen messages m_0 or m_1. However, this is more about the messages invisibility but may not protect the signer if the adversary is interested in knowing if the purported signer has participated or not instead (say Eve will agree to give Bob a job offer if Alice does, and how much salary Alice is going to offer Bob does not really matter to Eve). So here we consider the privacy requirement of the real signer, which ensures that $\mathcal{A}$ cannot distinguish whether a confirmer signature is signed using a signing key sk_0 or sk_1.

Definition 3 (Source Privacy)

$$
\begin{aligned}
|\ \Pr[\ & (\mathsf{xk}, \mathsf{pk}) \leftarrow \mathsf{EKGen}(1^\kappa); (m, (\mathsf{sk}_0, \mathsf{vk}_0), (\mathsf{sk}_1, \mathsf{vk}_1), state) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{E}}_{\mathsf{xk}}(\cdot)}(\mathsf{param}, \mathsf{pk}); \\
& b \leftarrow \{0,1\}; \sigma^* \leftarrow \mathsf{Sig}(\mathsf{sk}_b, m); \hat{\sigma}^* \leftarrow \mathsf{Des}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \sigma^*, m); \\
& b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{E}}_{\mathsf{xk}}(\cdot)}(state, \hat{\sigma}^*) : \\
& b = b' \mid \mathsf{Valid}(\mathsf{sk}_0, \mathsf{vk}_0) \wedge \mathsf{Valid}(\mathsf{sk}_1, \mathsf{vk}_1)] - \tfrac{1}{2} \mid < \ \mathsf{negl}(\kappa)
\end{aligned}
$$

where $\mathcal{O}^{\mathsf{E}}_{\mathsf{xk}}(\cdot)$ *is an extraction oracle, taking a valid confirmer signature* $\hat{\sigma} \neq \hat{\sigma}^*$ *as an input and returning the output of* $\mathsf{Ext}(\mathsf{xk}, \hat{\sigma})$.

One might think at first that the non-interactive confirmer signatures reveal too much information about the possible signer. However, while the two possible signers are known, they are equally probable as a real signer.

Note on Indistinguishability: Another way to model the privacy guarantee of the real signer is based on an indistinguishability notion. Specifically, there is a PPT algorithm Des' taking as inputs a message, the verification key of the purported signer, and possibly the signing and the verification key of a designated verifier, and outputting a fake/simulated signature such that it looks indistinguishable as a real signature generated by Des to any adversary $\mathcal{A}$. If the capability of extracting ordinary signature is present, $\mathcal{A}$ is disallowed to win in a trivial way, such as having the extraction key xk or asking the oracle to use xk to extract the signature out of the confirmer/designated-verifier signature in question.

We claim that our privacy notion implies indistinguishability. Our privacy notion guarantees that it is difficult to tell which key among $\{\mathsf{vk}_S, \mathsf{vk}_V\}$ is the actual verification key for the σ in $\mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \sigma, m)$. A faking/simulation algorithm Des' under our framework is readily available. We can do that by using the designated-verifier's signing key sk_V to create a signature $\sigma = \mathsf{Sig}(\mathsf{sk}_V, m)$ and then create a confirmer signature $\mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \sigma, m)$. This is exactly how a signature is created when the roles of the signer and the verifier are interchanged.

A similar argument is used in [6] to argue for *unconditional* indistinguishability in their case. While in our case, we added an encryption of the "real" verification key, so we can only achieve computational indistinguishability.

Stronger Privacy to Protect All Possible Signers' Identities: If one insists on hiding the verification keys in the designated-verifier signature, recall that it was shown in [12] that the notion of confirmer signatures is equivalent to public key encryption, so one may always add an extra layer of encryption and encrypt the whole signature under a public key of the designated verifier to achieve stronger privacy guarantee, without relying on any additional assumption.

Soundness. While privacy protects the signer, the verifier is protected by the soundness guarantee. Intuitively, the verifier does not want to get a confirmer signature $\hat{\sigma}$ which is valid according to the DVer algorithm, but the extracted ordinary signature of which cannot pass the Ver algorithm.

Definition 4 (Extraction Soundness)

$$\begin{aligned}\Pr[\ & (m,(\mathsf{sk}_0,\mathsf{vk}_0),(\mathsf{sk}_1,\mathsf{vk}_1),(\mathsf{xk},\mathsf{pk}),\hat{\sigma}) \leftarrow \mathcal{A}(\mathsf{param});(\sigma,b)=\mathsf{Ext}(\mathsf{xk},\hat{\sigma}) : \\ & \mathsf{DVer}(\{\mathsf{vk}_0,\mathsf{vk}_1\},\mathsf{pk},\hat{\sigma},m)=1 \wedge \mathsf{Ver}(\mathsf{vk}_b,\sigma,m)=0 \\ & \wedge \mathsf{Valid}(\mathsf{sk}_0,\mathsf{vk}_0) \wedge \mathsf{Valid}(\mathsf{sk}_1,\mathsf{vk}_1) \wedge \mathsf{Valid}(\mathsf{xk},\mathsf{pk})\ ] < \mathsf{negl}(\kappa)\end{aligned}$$

The extraction soundness only guarantees that a valid confirmer signature $\hat{\sigma}$ can be extracted to a valid ordinary signature created by either one of the secret keys. Since the creation of confirmer signature is universal in our scheme, it is possible that a designated verifier who expects to get someone else's signature eventually may end up with getting his own signature! An easy solution to this problem is to register a key pair which is only for the purpose of designation but not for signing any message. If a single key pair is used, a user should be cautious in what signature to accept and what to sign. For the offer scenario, it means that Bob should protect his private key secretly and refuse to sign any message similar to "xxx is willing to offer to Bob yyy", and should never get convinced by a confirmer signature on a message without mentioning who is agreed to make the offer, which should be a common practice to follow in checking an offer.

Unforgeability. There are two types of unforgeability to be considered. The first is the standard notion of unforgeability under chosen message attack. The second one similarly requires that it is infeasible to compute a confirmer signature $\hat{\sigma}$ on a new message which could be verified using DVer for a pair of verification keys, the secret keys of which are unknown. Note that Des is a public algorithm which requires no secret knowledge, so like in the regular unforgeability game the adversary needs access only to a signing oracle for the unknown signing keys.

Definition 5 (Unforgeability)

$$\begin{aligned}\Pr[\ & (\mathsf{sk}_0,\mathsf{vk}_0),(\mathsf{sk}_1,\mathsf{vk}_1) \leftarrow \mathsf{SKGen}(1^\kappa); \\ & (m^*,\sigma^*,\hat{\sigma}^*,(\mathsf{xk},\mathsf{pk})) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{S}}_{\mathsf{sk}_0}(\cdot),\mathcal{O}^{\mathsf{S}}_{\mathsf{sk}_1}(\cdot)}(\mathsf{param}); \\ & \rho=\mathsf{Ver}(\mathsf{vk}_0,\sigma^*,m^*);\hat{\rho}=\mathsf{DVer}(\{\mathsf{vk}_0,\mathsf{vk}_1\},\mathsf{pk},\hat{\sigma}^*,m^*) : \\ & m^* \notin \{m_i\} \wedge \mathsf{Valid}(\mathsf{xk},\mathsf{pk}) \wedge (\rho=1 \vee \hat{\rho}=1)\ ] < \mathsf{negl}(\kappa)\end{aligned}$$

where $\{m_i\}$ *is the set of messages supplied to* $\mathcal{O}^{\mathsf{S}}_{\mathsf{sk}_0}$ *or* $\mathcal{O}^{\mathsf{S}}_{\mathsf{sk}_1}$ *which take a message and output a digital signature signed with the secret key* sk_0 *and* sk_1*, respectively.*

Note that extraction soundness "converts" that any forgery of the second type to a forgery of the first type as the extracted signature has to be verifiable under one of the verification keys. One may also try to define unforgeability by allowing the adversary to generate $(\mathsf{sk}_1, \mathsf{vk}_1)$ and only considering $\hat{\sigma}^*$ is a valid forgery if one can extract a valid signature under vk_0 from $\hat{\sigma}^*$. However, this definition turns out to be equivalent to our unforgeability definition above.

3 Preliminaries

3.1 Number Theoretic Assumptions

Definition 6 (Bilinear Map). *Let* $\mathbb{G}$ *and* $\mathbb{G}_T$ *be two groups of prime order* p*. A bilinear map* $\hat{e}(\cdot,\cdot) : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *satisfies:*

- non-degeneracy: $\hat{e}(g,g)$ *is a generator of* $\mathbb{G}_T$ *when* g *is a generator of* $\mathbb{G}$*;*
- bilinearity: *for all* $x, y \in \mathbb{Z}_p$*,* $\hat{e}(g^x, g^y) = \hat{e}(g,g)^{xy}$*.*

Definition 7 (Decisional Linear (DLIN) Assumption [13]). *The DLIN assumption holds if for all PPT adversaries, on input a sextuple* $(u, v, g, u^a, v^b, g^c) \in \mathbb{G}^6$*, where* $c = a + b$ *or* c *is a random element in* $\mathbb{Z}_p$ *with equal probability, the probability of guessing which is the case correctly over* $\frac{1}{2}$ *is negligible.*

Definition 8 (q-Strong Diffie-Hellman (q-SDH) Assumption [14]). *The* q*-SDH assumption holds if for all PPT adversaries, it is of negligible probability to find a pair* $(m, g^{\frac{1}{m+x}}) \in \mathbb{Z}_p \times \mathbb{G}$ *when given* $(g, g^x, g^{x^2}, \ldots, g^{x^q}) \in \mathbb{G}^{q+1}$*.*

3.2 Cryptographic Building Blocks

Definition 9 (Collision-Resistant Hash Function (CRHF)). *A family of hash functions* $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{l(\kappa)}$ *is said to be collision resistant if for all PPT adversaries* $\mathcal{A}$*, we have:*

$$\Pr[H \in_R \mathcal{H};\ x, y \leftarrow \mathcal{A}(H) : H(x) = H(y)] < \mathsf{negl}(\kappa).$$

We use a CRHF to compress messages of any length to messages of length $l(\kappa)$, where $2^{l(\kappa)} < p$ and p is the order of the groups we are working with.

Definition 10 (Signature Schemes). *A signature scheme* Σ *is a triple of PPT algorithms* $(\mathsf{SKGen}, \mathsf{Sig}, \mathsf{Ver})$ *with the following properties:*

- SKGen*: takes as an input a security parameter* 1^κ*; outputs a signing key* sk *and a corresponding verification key* vk*.*
- Sig*: takes as inputs a signing key* sk *and a message* m *and outputs a signature* $\sigma = \mathsf{Sig}(\mathsf{sk}, m)$*.*

- Ver*: takes as inputs a verification key* vk*, a message* m *and a purported signature* σ*; outputs either* 1 *or* 0 *denoting "accept" or "reject".*
- *Correctness:* $\forall \kappa \in \mathbb{N}$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}(1^\kappa)$, $\mathsf{Ver}(\mathsf{vk}, \mathsf{Sig}(\mathsf{sk}, m), m) = 1$.

Definition 11 (Strong One-Time Signature Scheme). *A signature scheme* Σ *is said to be strongly secure one-time signature, if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr[\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}(); (m, state) \leftarrow \mathcal{A}(\mathsf{vk}); \sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, m); \\ (\sigma^*, m^*) \leftarrow \mathcal{A}(state, \sigma) : \mathsf{Ver}(\mathsf{vk}, \sigma^*, m^*) = 1 \ \wedge \ (\sigma^*, m^*) \neq (\sigma, m)] \ < \ \mathsf{negl}(\kappa).$$

In our construction, we could use any strong one-time signature scheme, preferably with security follows from some of the aforementioned assumptions. One such scheme is described in [15] which is based on the DLIN assumption. This scheme is also used in [16].

Definition 12 (Weakly-Secure Signatures). *A signature scheme* Σ *is defined to be secure against weak chosen message attack (wEUF-secure) if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr[\ (m_1, m_2, \ldots, m_q) \leftarrow \mathcal{A}(1^\kappa); (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}(); \\ \sigma_i = \mathsf{Sig}(\mathsf{sk}, m_i); (m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \sigma_1, \ldots, \sigma_q) : \\ \mathsf{Ver}(\mathsf{vk}, \sigma^*, m^*) = 1 \ \wedge \ m^* \notin \{m_1, \ldots, m_q\}\ ] \ < \ \mathsf{negl}(\kappa).$$

We use the signature scheme of Boneh and Boyen [14] which is wEUF-secure under the q-SDH assumption. The system parameters are $(p, \mathbb{G}, \mathbb{G}_T, g, \hat{e}(\cdot,\cdot), H(\cdot))$, where H is a collision-resistant hash function with range in $\mathbb{Z}_p$.

- $\mathcal{BB}.\mathsf{SKGen}()$: Pick $\mathsf{sk} \in_R \mathbb{Z}_p^*$ and compute $\mathsf{vk} = g^{\mathsf{sk}}$; output the pair of private signing key and the public verification key as $(\mathsf{sk}, \mathsf{vk})$.
- $\mathcal{BB}.\mathsf{Sig}(\mathsf{sk}, m)$: Output the signature $\sigma = g^{\frac{1}{\mathsf{sk}+H(m)}}$. (It fails to sign on m if $H(m) = -\mathsf{sk}$.)
- $\mathcal{BB}.\mathsf{Ver}(\mathsf{vk}, \sigma, m)$: Accept if and only if $\hat{e}(\sigma, \mathsf{vk} \cdot g^{H(m)}) = \hat{e}(g, g)$.

Tag-Based Encryption

Kiltz [17] extended the linear encryption [13] to a tag-based encryption which is secure against selective-tag weak chosen-ciphertext attacks (CCA), under the decision linear assumption.

- $\mathcal{TBE}.\mathsf{EKGen}(1^\kappa)$: The encryption key is $(u, v, g_0, U, V) \in \mathbb{G}^5$ where $u^a = v^b = g_0$, and the decryption key is (a, b).
- $\mathcal{TBE}.\mathsf{Enc}((u, v, g_0, U, V), m, \ell)$: To encrypt a message $m \in \mathbb{G}$ under a tag (or a label) $\ell \in \mathbb{Z}_p^*$, picks $\varphi, \psi \in_R \mathbb{Z}_p^*$ and returns $(T_1, T_2, T_3, T_4, T_5) = (u^\varphi, v^\psi, m g_0^{\varphi+\psi}, (g_0^\ell U)^\varphi, (g_0^\ell V)^\psi)$,
- $\mathcal{TBE}.\mathsf{Dec}((a, b), (T_1, T_2, T_3, T_4, T_5), \ell)$: To decrypt $(T_1, T_2, T_3, T_4, T_5)$, return $T_3/(T_1^a \cdot T_2^b)$ if $\hat{e}(u, T_4) = \hat{e}(T_1, g_0^\ell U)$ and $\hat{e}(v, T_5) = \hat{e}(T_2, g_0^\ell V)$ hold. The latter check can also be done without pairing if the discrete logarithm of U, V with respect to u, v respectively are kept as part of the private key.

The message space of this encryption scheme is $\mathbb{G}$, which matches with the signature space as well as the verification key space of the signature scheme $\mathcal{BB}$.

Non-interactive Proofs for Bilinear Groups

Groth and Sahai [18] developed techniques for proving statements expressed as equations of certain types. Their proofs are non-interactive and in the common references string (CRS) model. The proofs have perfect completeness and, depending on the CRS, perfect soundness or witness indistinguishability / zero-knowledge. The two types of CRS are computationally indistinguishable. Groth and Sahai showed how to construct such proofs under various assumptions, one of which is the decisional linear assumption.

The first type of equations were interested in is linear equations over $\mathbb{G}$ (described as multi-exponentiation of constants in the one sided case in [18]), of the form $\prod_{j=1}^{L} a_j^{\chi_j} = a_0$, where $\chi_1, \ldots \chi_L$ are variables and $a_0, a_1, \ldots, a_L \in \mathbb{G}$ are constants. Such equations allow to prove equality of committed and encrypted values, with the randomness used to commit and encrypt being the witness (the assignment of the variables) which satisfies the corresponding equations. The proofs for this type of equations are zero-knowledge, i.e. valid proofs could be produced without a witness using the trapdoor of the simulated CRS.

Pairing product equations allow to prove validity of $\mathcal{BB}$ signatures without revealing the signature and/or the verification key, i.e., $\hat{e}(\sigma, v \cdot g^m) = \hat{e}(g, g)$ for variables σ and v. The Groth-Sahai proofs for this type of equations are only witness indistinguishable, which is sufficient for our purposes, though could be transformed into zero-knowledge proofs if certain requirements are met like in the case of the above equation.

The last type of equations we need is to assert the plaintext of an encryption $\mathfrak{C}$ is one of two publicly known messages m_1 and m_2. The is the key step for the designated verification. Rather than using OR-proofs which do not mesh well with Groth-Sahai proofs, we introduce two additional variables α, β to be used in the exponent for which we prove $\alpha + \beta = 1$, both α and $\beta \in \{0, 1\}$, and the ciphertext $\mathfrak{C}$ being an encryption of $m_1^{\alpha} m_2^{\beta}$. The first proof is done using the linear equation $g^{\alpha} g^{\beta} = g$; $\alpha, \beta \in \{0, 1\}$ is proven using the technique of Groth *et. al.* [19] which constructs a witness-indistinguishable proof for a commitment of $\chi \in \mathbb{Z}_p$ being a commitment of 0 or 1; and that $\mathfrak{C}$ contains the proper plaintext is shown using linear equations.

4 Our Construction

4.1 High Level Idea

We present an efficient non-interactive confirmer signature scheme using the tools we just described. The key property is that our confirmer signature is publicly verifiable, but its holder cannot tell whether it was produced by the signer or the designated verifier (unless, of course, in possession of the extraction key). One elegant way to make this possible is that we create the designated-verifier signature by proving the statement "*θ is a valid signature signed by either Alice or Bob*". While it is essentially the concept of ring signature [20], it is the confirmer who performs the designation in our case, but not the signer. We stress

that, in existing applications of using a 2-user ring signature as a designated verifier signature, *the signer* is the one who performs the designation. In other words, the universal confirmer has the ability to "turn" *any* ordinary signature into a 2-user ring signature by involving any designated verifier in the "ring".

To make this proof efficient and non-interactive, we employ the NIZK/NIWI proof system proposed by Groth and Sahai [18]. This is a powerful primitive but the target language is limited. Hence, building blocks for the construction are chosen so as to be within this limit. For the signature scheme, we combine a strong one-time signature and a weak $\mathcal{BB}$ signature [14] we reviewed earlier, the latter of which is convenient when constructing efficient non-interactive proofs. The $\mathcal{BB}$ signature is used to sign random messages, i.e., verification keys for the one-time signature scheme, so security against weak chosen message attack suffices, we then use the one-time signature scheme to sign on the actual message.

To realize our universal designation, the designation algorithm makes commitments for the verification key and the $\mathcal{BB}$ signature, proves those committed values satisfy the verification equation with the message being the one-time verification key, and proves that the verification key commitment is either of the signer's or of the designated verifier's verification key.

To achieve extractability of ordinary signature and satisfy our privacy requirement, we require a CCA2-secure public-key encryption. The encryption scheme should be chosen so we could prove equality of committed and encrypted values using the Groth-Sahai proofs. Two appropriate schemes are the tag-based encryption scheme of Kiltz [17], and Shacham's Linear Cramer-Shoup [21]. Proving that the plaintext of an encryption is one of two possible messages require some extra work as we described in Section 3.2.

To prevent the adversary from massaging the confirmer signature and learn (partial) information of the real signer or the encrypted signature, we employ a public key encryption scheme with label and another one-time signature to ensure the non-malleability of the confirmer signature.

The verification and designated verification are straightforward – simply checking the validity of the signatures/proofs. The extraction algorithm uses the secret key of the encryption scheme to decrypt the $\mathcal{BB}$ signature and the verification key from the corresponding ciphertexts.

4.2 Instantiation

– $\mathsf{Setup}(1^\kappa)$: This setup algorithm takes up a bit string 1^κ, picks groups $\mathbb{G}, \mathbb{G}_T$ of prime order p with a bilinear map $\hat{e}(\cdot,\cdot)$, where $2^\kappa < p < 2^{\kappa+1}$. This bilinear map context determines the common reference string (CRS) for the Groth-Sahai proof system, which in turn determines the Groth-Sahai commitment function $\mathsf{Com}(m; \boldsymbol{r})$ which commits to $m \in \mathbb{G}$ or $m \in \mathbb{Z}_p$ using appropriately sampled randomness vector $\boldsymbol{r}$. This algorithm also chooses a collision resistant hash function $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{Z}_p$.

  All these parameters are concatenated into a single string param. For brevity, we omit the inclusion of param in the interface of our algorithms, which makes some of the algorithms like $\mathsf{EKGen}()$ has no explicit input.

- $\mathsf{SKGen}()$: $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathcal{BB}.\mathsf{SKGen}()$.
- $\mathsf{Sig}(\mathsf{sk}_S, m)$:
  1. $(\mathsf{osk}, \mathsf{ovk}) \leftarrow \mathcal{OTS}.\mathsf{SKGen}()$.
  2. If $H(\mathsf{ovk}) = -\mathsf{sk}_S$, repeat step 1.
  3. Output $\sigma = (\mathsf{ovk}, \sigma_{bb} = \mathcal{BB}.\mathsf{Sig}(\mathsf{sk}_S, H(\mathsf{ovk})), \sigma_{ots} = \mathcal{OTS}.\mathsf{Sig}(\mathsf{osk}, m))$.
- $\mathsf{Ver}(\mathsf{vk}, \sigma = (\mathsf{ovk}, \sigma_{bb}, \sigma_{ots}), m)$: Output 1 if and only if $\mathcal{BB}.\mathsf{Ver}(\mathsf{vk}, \sigma_{bb}, H(\mathsf{ovk}))$ and $\mathcal{OTS}.\mathsf{Ver}(\mathsf{ovk}, \sigma_{ots}, m)$ both output 1; otherwise, output 0.
- $\mathsf{EKGen}()$: $(\mathsf{xk}, \mathsf{pk}) \leftarrow \mathcal{TBE}.\mathsf{EKGen}()$.
- $\mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \sigma = (\mathsf{ovk}, \sigma_{bb}, \sigma_{ots}), m)$:
  - Initialization:
    1. $(\mathsf{osk}', \mathsf{ovk}') \leftarrow \mathcal{OTS}.\mathsf{SKGen}()$.
    2. Order $\{\mathsf{vk}_S, \mathsf{vk}_V\}$ lexicographically into $(\mathsf{vk}_0, \mathsf{vk}_1)$.
  - Commit and encrypt the verification key and prove their well-formedness:
    1. Encrypt vk_S labelled with ovk' in $\mathfrak{C}_{\mathsf{vk}} \leftarrow \mathcal{TBE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{vk}_S, \boldsymbol{r'}, H(\mathsf{ovk}'))$.
    2. Create π_{enc} which is a proof that $\mathfrak{C}_{\mathsf{vk}}$ is an encryption of $\mathsf{vk}_S = \mathsf{vk}_0^{\alpha}\mathsf{vk}_1^{\beta}$ using NIZK proofs for satisfiable linear equations with variables $\boldsymbol{r'}, \alpha, \beta$, with proofs for $\alpha + \beta = 1$ and $\alpha, \beta \in \{0, 1\}$.
    3. Create a commitment of the verification key by $C_{\mathsf{vk}} = \mathsf{Com}(\mathsf{vk}_S;\ \boldsymbol{r})$.
    4. Create π_{vk} which is a proof of equality of the committed/encrypted values in C_{vk} and $\mathfrak{C}_{\mathsf{vk}}$ using NIZK proofs for satisfiable linear equations with variables $\boldsymbol{r'}, \boldsymbol{r}$.
  - Commit and encrypt the key-certifying signature and prove their well-formness:
    1. Encrypt σ_{bb} labelled with ovk' in $\mathfrak{C}_{\sigma} \leftarrow \mathcal{TBE}.\mathsf{Enc}(\mathsf{pk}, \sigma_{bb};\ \boldsymbol{s'}, H(\mathsf{ovk}'))$.
    2. Create a commitment of the signature by $C_{\sigma} = \mathsf{Com}(\sigma_{bb};\ \boldsymbol{s})$.
    3. Create π_{σ} which is a proof of equality of the committed/encrypted values in C_{σ} and $\mathfrak{C}_{\sigma}$ using NIZK proofs for satisfiable linear equations with variables $\boldsymbol{s'}, \boldsymbol{s}$.
  - Linking all pieces together:
    1. Create π_{sgn} which is an NIWI proof of validity of the $\mathcal{BB}$ signature for the committed values of C_{vk} and C_{σ}; C_{vk} and C_{σ} are commitments produced by the proof system to create π_{sgn} but are given explicitly in the construction as we require equality of committed values (used for the proofs) and encrypted ones (used for extraction).
    2. During the creation of the above proofs, commitments of the variables $\boldsymbol{r'}, \boldsymbol{r}, \boldsymbol{s'}, \boldsymbol{s}$ are also created. Let π be this set of the proofs $\pi_{enc}, \pi_{vk}, \pi_{\sigma}, \pi_{sgn}$ and the associated commitments. Also, let $m' = (\mathsf{ovk}', \mathsf{vk}_0, \mathsf{vk}_1, \mathsf{ovk},\ \sigma_{ots},\ \mathfrak{C}_{\mathsf{vk}},\ \mathfrak{C}_{\sigma},\ \pi)$.
    3. Sign on the string m' by $\hat{\sigma}' \leftarrow \mathcal{OTS}.\mathsf{Sig}(\mathsf{osk}', m')$.
    4. Output $\hat{\sigma} = (\hat{\sigma}', m')$.
- $\mathsf{DVer}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \hat{\sigma} = (\hat{\sigma}', m'), m)$: Verify the one-time signatures $\hat{\sigma}'$ on m' under ovk' and all the NIZK/NIWI proofs; also check that $\{\mathsf{vk}_0, \mathsf{vk}_1\}$ are the same verification keys (after ordering them lexicographically) as those in m'. Finally, verify the one-time signature σ_{ots} on m under ovk. If any of the verification is not successful, return 0; otherwise, return 1.

– $\mathsf{Ext}(\mathsf{xk}, \hat{\sigma} = (\hat{\sigma}', m'))$:
  1. Parse m' as $(\mathsf{ovk}', \mathsf{vk}_0, \mathsf{vk}_1, \mathsf{ovk},\ \sigma_{ots},\ \mathfrak{C}_{\mathsf{vk}},\ \mathfrak{C}_{\sigma},\ \pi)$.
  2. Decrypt $\mathfrak{C}_{\sigma}$ to get $\sigma_{bb} = \mathcal{TBE}.\mathsf{Dec}(\mathsf{xk}, \mathfrak{C}_{\sigma}, H(\mathsf{ovk}'))$.
  3. Decrypt $\mathfrak{C}_{\mathsf{vk}}$ to get $\mathsf{vk} = \mathcal{TBE}.\mathsf{Dec}(\mathsf{xk}, \mathfrak{C}_{\mathsf{vk}}, H(\mathsf{ovk}'))$.
  4. If any decryption was rejected or $\mathcal{OTS}.\mathsf{Ver}(\mathsf{ovk}', m') = 0$, output $(\perp, \perp)$.
  5. Output $((\mathsf{ovk}, \sigma_{bb}, \sigma_{ots}), b)$ where $\mathsf{vk}_b = \mathsf{vk}$.

In the the full version, we prove the following theorem:

Theorem 13. *The described non-interactive confirmer signature scheme is secure under the Decisional Linear assumption and the q-SDH assumption, assuming the hash function is collision resistant. That is, the scheme satisfies the correctness, privacy, soundness, and unforgeability requirements.*

4.3 Discussion

All the proofs used in our scheme can be done by variants of the proofs in some existing cryptosystems involving Groth-Sahai proofs as mentioned in Section 3.2. Basically, our confirmer signature is proving that an encrypted signature is a valid one under an encrypted public key, where the public key comes from one of two possibilities. The two possibilities part involves an OR proof as discussed in Section 3.2. The encryption part involves proving that the plaintext of the encryption and the committed value in the corresponding commitment are the same. The proofs of the latter kind for the encryption scheme $\mathcal{TBE}$ has appeared in existing group signature schemes (e.g. [22, Section 7]). With these proofs, the rest is about proving the signature in the commitment verifies, and the corresponding proof for the signature scheme $\mathcal{BB}$ involves a simple pairing product equation which can be found in many "privacy-oriented" Groth-Sahai-proof-based cryptosystems such as anonymous credential (e.g. [23, Section 5]) and group signatures (e.g. [22, Section 7]). Due to the space limitation, we defer the details on the language or the equation required to the full version.

There is an important difference regarding the usage of the signature scheme in the aforementioned systems [23,22] and in our scheme. For the latter, the signature scheme is often used in certifying a user public key by the private key of the "authority". Obviously, the verification of a valid credential/signature would not require the knowledge of the user private key, and hence the signature is essentially signing on a user private key which is "hidden" in the output of a certain one-way function. On the other hand, we just use the signature scheme to sign on a public one-time signature verification key. This is the reason why we do not consider a possible stronger notion such as F-unforgeability [23].

Regarding efficiency, each signature consists of roughly 100 group elements, while the scheme of Liskov-Micali [3] produces signatures with $O(\kappa)$ ciphertexts.

5 Concluding Remarks

We unify the concept of confirmer signatures and designated-verifier signatures. Specifically, we introduce the notion of *non-interactive* confirmer signatures,

which can also be interpreted as *extractable* universal designated-verifier signatures. Besides saving in valuable rounds of interaction, we believe a non-interactive construction of confirmer signatures represents a more natural instantiation of the primitive. Most importantly, it resolves the problem of on-line transferability [3] when the recipient is acting as a man-in-the-middle, in a simple and computationally-efficient way. Our proposed construction is a proof-of-concept scheme. There are many possibilities for optimization. For examples, one may improve our construction by picking better underlying primitives, or try to get rid of using encryption by leveraging the strong unforgeability [24]. Finally, for practical application, one may consider resorting to random oracle model and propose a possibly more efficient implementation.

References

1. Chaum, D., Antwerpen, H.V.: Undeniable Signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
2. Chaum, D.: Designated Confirmer Signatures. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 86–91. Springer, Heidelberg (1995)
3. Liskov, M., Micali, S.: Online-untransferable signatures. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 248–267. Springer, Heidelberg (2008)
4. Gentry, C., Molnar, D., Ramzan, Z.: Efficient Designated Confirmer Signatures Without Random Oracles or General Zero-Knowledge Proofs. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 662–681. Springer, Heidelberg (2005)
5. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
6. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal Designated-Verifier Signatures. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 523–542. Springer, Heidelberg (2003)
7. Baek, J., Safavi-Naini, R., Susilo, W.: Universal Designated Verifier Signature Proof (or How to Efficiently Prove Knowledge of a Signature). In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 644–661. Springer, Heidelberg (2005)
8. Shahandashti, S.F., Safavi-Naini, R., Baek, J.: Concurrently-Secure Credential Ownership Proofs. In: ASIACCS, pp. 161–172 (2007)
9. Boyar, J., Chaum, D., Damgård, I., Pedersen, T.P.: Convertible Undeniable Signatures. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 189–205. Springer, Heidelberg (1991)
10. Aimani, L.E.: Toward a Generic Construction of Universally Convertible Undeniable Signatures from Pairing-Based Signatures. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 145–157. Springer, Heidelberg (2008)
11. Aimani, L.E.: On Generic Constructions of Designated Confirmer Signatures. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 343–362. Springer, Heidelberg (2009)
12. Okamoto, T.: Designated Confirmer Signatures and Public-Key Encryption are Equivalent. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 61–74. Springer, Heidelberg (1994)
13. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

14. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
15. Groth, J.: Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
16. Camenisch, J., Chandran, N., Shoup, V.: A Public Key Encryption Scheme Secure against Key Dependent Chosen Plaintext and Adaptive Chosen Ciphertext Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)
17. Kiltz, E.: Chosen-Ciphertext Security from Tag-Based Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
18. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
19. Groth, J., Ostrovsky, R., Sahai, A.: Perfect Non-interactive Zero Knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
20. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret: Theory and Applications of Ring Signatures. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Theoretical Computer Science. LNCS, vol. 3895, pp. 164–186. Springer, Heidelberg (2006)
21. Shacham, H.: A Cramer-Shoup Encryption Scheme from the Linear Assumption and from Progressively Weaker Linear Variants. Cryptology ePrint Archive, Report 2007/074 (2007), `http://eprint.iacr.org/`
22. Groth, J.: Fully Anonymous Group Signatures Without Random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
23. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and Noninteractive Anonymous Credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
24. Schuldt, J.C.N., Matsuura, K.: An Efficient Convertible Undeniable Signature Scheme with Delegatable Verification. In: Kwak, J., Deng, R.H., Won, Y., Wang, G. (eds.) ISPEC 2010. LNCS, vol. 6047, pp. 276–293. Springer, Heidelberg (2010)