Efficient Designated Confirmer Signatures Without Random Oracles or General Zero-Knowledge Proofs (Extended Abstract)

Craig Gentry[1], David Molnar[2] and Zulfikar Ramzan[1]

[1] DoCoMo USA Labs, {cgentry,ramzan}@docomolabs-usa.com
[2] University of California, Berkeley dmolnar@eecs.berkeley.edu

Abstract. Most prior designated confirmer signature schemes either prove security in the random oracle model (ROM) or use general zero-knowledge proofs for NP statements (making them impractical). By slightly modifying the definition of designated confirmer signatures, Goldwasser and Waisbard presented an approach in which the Confirm and ConfirmedSign protocols could be implemented without appealing to general zero-knowledge proofs for NP statements (their Disavow protocol still requires them). The Goldwasser-Waisbard approach could be instantiated using Cramer-Shoup, GMR, or Gennaro-Halevi-Rabin signatures.

In this paper, we provide an alternate generic transformation to convert any signature scheme into a designated confirmer signature scheme, without adding random oracles. Our key technique involves the use of a signature on a commitment and a separate encryption of the random string used for commitment. By adding this "layer of indirection," the underlying protocols in our schemes admit efficient instantiations (i.e., we can avoid appealing to general zero-knowledge proofs for NP statements) and furthermore the performance of these protocols is not tied to the choice of underlying signature scheme. We illustrate this using the Camenisch-Shoup variation on Paillier's cryptosystem and Pedersen commitments. The confirm protocol in our resulting scheme requires 10 modular exponentiations (compared to 320 for Goldwasser-Waisbard) and our disavow protocol requires 41 modular exponentiations (compared to using a general zero-knowledge proof for Goldwasser-Waisbard). Previous schemes use the encryption of a signature paradigm, and thus run into problems when trying to implement the confirm and disavow protocols efficiently.

1 Introduction

Digital signatures allow one party to sign a message and have the resulting message-signature pair be verifiable by anyone. There are, however, situations when the signer may want to limit signature recipient's ability to present the signature to others. Chaum and van Antwerpen [9] introduced the notion of *Undeniable Signatures* to help achieve this aim. Such signatures require the involvement of the original signer each time they are verified. Of course, this approach places excessive responsibility on the signer to be available and to not act maliciously. As an alternate approach, Jakobsson et al. [18] introduced *Designated Verifier* signatures which can only be validated by a specific user (who the signer can designate). However, the recipient cannot convince another party of the signature's validity, which is a desirable feature in many situations.

Chaum [7] introduced the notion of a *designated confirmer signature* (DCS), which overcomes both these limitations. Such signatures require the assistance of a trusted third party called the *confirmer*. Given a signature σ that the signer issues, the confirmer can execute a special $\mathsf{Confirm}$ protocol to prove that a signature σ is a valid signature on a message m, or a $\mathsf{Disavow}$ protocol to show that σ is not a valid signature on a message m. Without the confirmer, however, no party can determine whether σ is a valid signature for m or not.

Applications of Designated Confirmer Signatures. Designated confirmer signatures have several cryptographic applications. For example, Asokan et al. demonstrate their use for "optimistic" fair exchange, in which a trusted third party need be involved only if cheating is suspected; the third party plays the role of confirmer [1]. Chaum notes that they are useful for integrity-critical content, such as virus updates, which is updated as part of a subscription service. In this application, without the assistance of the confirmer (for a fee), the content cannot be verified [7]. Therefore, efficient designated confirmer schemes are of practical as well as theoretical interest.

Definitional Issues in DCS Schemes. Okamoto provided the first formal definition of a designated confirmer signature [21]. His scheme is simple to describe: use an ordinary signature scheme to sign a message, then use an ordinary public-key encryption scheme to encrypt the resulting signature with the designated confirmer's public key. General zero-knowledge proofs for NP statements can be used to provide the $\mathsf{Confirm}$ and $\mathsf{Disavow}$ protocols. Of course such proofs involve a reduction step to an NP-complete language (e.g., the language representing graphs that are three colorable), and cannot really be used in practice. Work on designated confirmer schemes continued, and it became clear that the correct definition had numerous subtleties, involving issues including coercion of the signer, handling multiple signers with the same confirmer, and others. We focus on two key issues in the definition of designated confirmer signature schemes that will affect our construction. Firstly, we must consider whether to require designated confirmer signatures to be simulatable. Michels and Stadler, following Okamoto, adopt a definition that requires "invisibility" of designated confirmer signatures [26, 21]; i.e., there exists a simulator who can create a "valid-looking" designated confirmer signature for any message m. Secondly, we must consider whether the $\mathsf{Confirm}$ and $\mathsf{Disavow}$ protocols are required to be zero-knowledge or instead may satisfy some weaker property. Previous definitions, such as those of Michels and Stadler, have included both these requirements [26]. Subsequent designated confirmer schemes were proposed by Camenisch and Michels [4], and by Camenisch and Shoup [6]. Other than the Okamoto scheme just mentioned, the remaining schemes all have security proofs in the random oracle model [2].

Goldwasser and Waisbard were interested in coming up with schemes that were provably secure without appealing to random oracles and without appealing to generic zero-knowledge proofs [17]. They found a clever way to approach this goal. They first argued that the previous definitions are too strong and do not capture the motivating applications of designated confirmer signature schemes. They modified Okamoto's original definition to require only that an adversary cannot verify signatures unless specifically assisted by the designated confirmer. Following the definition, they then show constructions of designated confirmer signatures for some specific signature schemes using strong *witness hiding proofs of knowledge* as a tool.

While Goldwasser and Waisbard can completely circumvent the use of random oracles, their Disavowal protocol still requires a generic zero-knowledge proof. Also, their witness hiding proofs of knowledge have soundness error $\frac{1}{2}$ and so are repeated λ times, where λ is the security parameter, to reduce the error. The efficiency of their protocols depends on the specific signature scheme. For example, with Cramer-Shoup as the underlying signature scheme each repetition costs 2 exponentiations in the underlying group [17].

Confirmer Commitments Revisited. Our approach is to use a Paillier-based instantiation of the confirmer commitment idea of Michels and Stadler [26]. A *confirmer commitment* is a special type of commitment scheme with a public key. Anyone can use the public key to create a commitment to a message m, but only the holder of the corresponding secret key can prove or disprove the resulting commitment is a commitment to the message m.

Michels and Stadler noted that signing a confirmer commitment would be an "obvious" approach to building a designated confirmer signature scheme [26]. This is because the confirmer commitment can only be matched to a message m by the confirmer, who holds the secret key. Unfortunately, their definition required "invisibility" for confirmer signatures, namely that given any message, there exists a simulator that can produces a "valid-looking" designated confirmer signature on that message. This ruled out the approach *a priori*, because signatures from the underlying signature scheme cannot be simulated for an existentially unforgeable signature scheme; any such simulator would imply the existence of a forger. As Goldwasser and Waisbard have argued, however, such requirements are too strong when we only care that no adversary can convince others that a signature is valid. We show in this paper that replacing the simulation requirement with the more natural non-verifiability of signatures requirement allows us to prove that the "obvious" scheme is secure.

We stress that, by itself, this observation is *not* enough. A more serious problem with the approach of confirmer commitments, as shown by Camenisch and Michels [4], is that the construction of confirmer commitments proposed by Michels and Stadler is actually not secure when multiple signers share the same confirmer [26]. Therefore it is not trivial to create a secure designated confirmer signature scheme from confirmer commitments, even with the weakened definition. While Camenisch and Michels sketch a fix for the Michels and Stadler construction, they do not give a full proof of security. Finally, while Camenisch and Michels define a generic transformation and prove it secure, efficient instantiation depends on the details of the underlying signature scheme; we analyze their suggested instantiation, which uses RSA with Full Domain Hash, in Section 5 [4]. We give a transformation in Section 4 that achieves security and efficiency given any signature scheme existentially unforgeable against chosen message attack. Instead of giving a new definition for confirmer commitments, however, we simply prove directly in Section 4 that the resulting scheme is a secure designated confirmer scheme.

Camenisch and Shoup's Approach. Camenisch and Shoup introduced new protocols for verifiable encryption and proving inequality of discrete logarithms [6]. Goldwasser and Waisbard note that these protocols are of interest in the designated confirmer setting [17]. Indeed, Camenisch and Shoup sketch a construction of designated confirmer signatures as part of their work on verifiable encryption. While we will use the protocols of Camenisch and Shoup in our construction, the way in which we use them is different. We sign a commitment and then encrypt information needed to open the commitment, while Camenisch and Shoup perform a verifiable encryption of a signature directly.

Because the Camenisch-Shoup approach encrypts a signature directly, the efficiency of Confirm and Disavow depends essentially on the underlying signature scheme. Their sketch suggests the use of Schnorr signatures [24], together with a trick for reducing the validity of a Schnorr signature to an equality of discrete logarithms. Details of the resulting scheme are not given, but are due to appear in a forthcoming paper. Since this paper is not yet available, in Section 5 we flesh out what we suspect the details to be to allow for comparison between our approach and theirs.

We highlight one such detail that affects our construction of DCS schemes: the Camenisch-Shoup protocols as described satisfy only special honest-verifier zero-knowledge. Special honest-verifier is not a strong enough property in the context of an adversary for DCS schemes, because the DCS adversary may act as a cheating verifier during Confirm and Disavow protocols. We show how to use techniques of Cramer, Damgard, and MacKenzie to fix this problem [13].

Finally, we note that the Camenisch-Shoup sketch suggests the use of Schnorr signatures, which require random oracles. Adapting their approach to a new signature scheme without random oracles is not trivial. We briefly outline difficulties with several such schemes in Section 5. Our approach, in contrast, works for any signature scheme without modification and easily yields a designated confirmer signature that does not need random oracles.

Our contributions. We make three contributions. First, we show that adopting the Goldwasser-Waisbard definition [17] allows us to prove that the approach of signing a commitment yields a designated confirmer signature scheme using any secure signature scheme as a building block. We specify a generic construction (based on any signature scheme, encryption scheme, and commitment scheme) and prove it secure *without* the use of random oracles. Second, we give an instantiation using Pedersen Commitments [23] together with Camenisch and

	Random Oracle	Signature	Confirm	Disavow
GW [17]	No	Cramer-Shoup [10]	2λ exps	generic ZKP
GW [17]	No	GMR [16]	2λ exps	generic ZKP
GW [17]	No	GHR [14]	2λ exps	generic ZKP
CS [6]	Yes	Schnorr [24]	10 exps	41 exps
CM [4]	Yes	RSA-FDH	24λ exps	60λ exps
Our scheme	No	Any	10 exps	41 exps

Fig. 1. Comparison of our approach to Goldwasser-Waisbard, Camenisch-Shoup, and Camenisch-Michels. Here λ is a security parameter. We achieve efficient $\mathsf{Confirm}$ and $\mathsf{Disavow}$ protocols without using random oracles. Section 5 explains these results in detail.

Shoup's [6] variant of Paillier's cryptosystem [22]. This approach achieves $\mathsf{Confirm}$ and $\mathsf{Disavow}$ protocols without appealing to generic zero-knowledge proofs and without appealing to random oracles independent of the choice of signature scheme. The resulting $\mathsf{Confirm}$ protocol requires 5 exponentiations (compared to 320 for Goldwasser-Waisbard) and our $\mathsf{Disavow}$ protocol requires 17,000 modular multiplications (whereas Goldwasser-Waisbard require a potentially very expensive generic zero-knowledge proof). Of course, we base our security on the security of Paillier's cryptosystem and on the security of Pedersen commitments, whereas Goldwasser and Waisbard require different assumptions than we do; we elaborate on this, and other aspects of the comparison, in Section 5. Third, we show that the resulting $\mathsf{Confirm}$ and $\mathsf{Disavow}$ protocols are zero-knowledge (whereas Goldwasser-Waisbard provide strong witness hiding proofs of knowledge), even against cheating verifiers, by combining the Camenisch-Shoup protocols with techniques of Cramer, Damgard and MacKenzie [13].

2 Preliminaries

Throughout λ is a positive integer denoting the security parameter. The security parameter is an implicit input to the algorithms discussed throughout the paper (and we omit it from the list of inputs when it might be otherwise clear from context). Let $\mathsf{negl}(\lambda)$ denote a negligible function; i.e., one that grows smaller than $1/\lambda^c$ for all c and all sufficiently large λ. For a positive integer a, we let $[a]$ denote the set $\{0, \ldots, a-1\}$. If $\mathsf{Alg}(\cdot, \cdot, \ldots)$ is a probabilistic algorithm, then $\mathsf{Alg}(x_1, x_2, \ldots)$ is a probability space over the random choices made by Alg. We let $x \leftarrow \mathsf{Alg}(x_1, x_2, \ldots)$ denote the experiment of running Alg on inputs $x_1, x_2, \ldots$, where x is a discrete random variable denoting the outcome. Note that Alg implicitly induces a distribution on the possible outputs. For a set S, we let $|S|$ denote the number of elements in S. If S is defined by a mathematical group, then $|S|$ is the group order. If S is equipped with a probability distribution D, we let $x \overset{D}{\longleftarrow} S$ denote the experiment of choosing $x \in S$ according to D. We typically let the underlying distribution be the uniform distribution R. In our context an adversary (denoted by $\mathcal{A}$, $\mathcal{F}$, etc. depending on the situation) is a probabilistic polynomial time random access machine with oracle access to some number κ of oracles, each of which is capable of computing some specified function. If $(f_1, \ldots, f_\kappa)$ is a κ-tuple of (oracle implemented) functions, and $\mathcal{A}$ is a κ-oracle adversary, then $\mathcal{A}^{f_1,\ldots,f_\kappa}$ denotes the adversary augmented with its oracles. An adversary may also have oracle access to a *protocol*, in which case the adversary provides protocol inputs from one side of the protocol (e.g., a verifier in an interactive proof system) and has oracle access to the responses from the other side of protocol (e.g., a prover in an interactive proof system). As above, the adversary on a given set of inputs induces a probability space, and we can associate a discrete random variable to the output of the experiment of running the adversary on a given set of inputs equipped with a given set of oracles. We now describe some of the tools required for our construction.

Proofs of Knowledge. Some of our protocols will be proofs of knowledge (PoK), as defined by Bellare and Goldreich [3]. Informally, an interactive proof (P, V) for a relation $R = \{\alpha, \beta\}$ is a proof of knowledge if there exists a probabilistic polynomial time knowledge extractor E who can extract a witness β given oracle access to a (possibly cheating) prover. The knowledge extractor is allowed to rewind the prover if necessary. The *knowledge error* of a ZKPoK quantifies the success probability of the extractor in terms of the prover's probability of convincing the verifier. Specifically, let P be a prover with respect to α. We say that the proof of knowledge has knowledge error $\kappa(\alpha)$ if, when the prover succeeds with probability $\epsilon(\alpha)$ the knowledge extractor E^P succeeds with probability at least $\epsilon(\alpha) - \kappa(\alpha)$.
Zero-Knowledge Proofs. The well-known Chaum-Pedersen protocol for proving equality of discrete logarithms and the Camenisch-Shoup protocols (which we describe below), are *special honest-verifier zero-knowledge Σ-protocols* [19] (SHVZK). This means that the protocols are public-coin and can be simulated assuming an honest verifier (i.e., a verifier that picks challenges uniformly at random). The *special* property means there is a simulator that, given the challenge of a verifier, can create the prover's messages.

The zero-knowledge proofs in our DCS scheme, however, must be zero-knowledge even against arbitrarily cheating verifiers. Moreover, we must be careful that our ZK proofs in our scheme not only reveal nothing about the witness, but also that the transcripts of a "real-world" interaction between the prover and a verifier in the ZK proof are indistinguishable from a transcript that a probabilistic polynomial-time simulator can generate using rewindable black-box access to the verifier. Such ZK proof protocols can be found in [15, 13]. For our efficient instantiation, we prefer the Cramer-Damgard-Mackenzie (CDM) approach, in which a prover P proves knowledge of a witness w for x to verifier V using SHVZK Σ-protocols in both directions, roughly as follows:

- Part 1: V commits to a value e and proves knowledge of e;
- Part 2: P gives a witness-indistinguishable proof of knowledge of either the verifier's value e or the witness w.

When applied to SHVZK 3-round proofs of knowledge, this approach yields a zero-knowledge proof of knowledge (ZKPoK) with negligible knowledge error, neglible soundness error, and which remains perfect zero-knowledge even against malicious adversaries. Specifically, there exists a simulator $S^{V'}$ that, given access to an arbitrary verifier V', outputs a transcript identically distributed to the transcript of interactions between the prover and V'. We will use this simulator extensively in the reduction algorithm of our DCS scheme's proof of security. Strictly speaking, the SHVZK 3-round proof of knowledge must satisfy an additional condition, namely that an associated "commitment relation" also have a 3-round proof of knowledge. Fortunately, as we will see, this is the case for our efficient instantiation, because we can leverage an efficient discrete logarithm protocol given by Cramer et al. [13]

Camenisch-Shoup Verifiable Encryption. We use an adaptation of Camenisch and Shoups's Paillier-based encryption scheme, which allows verifiable encryption of discrete logarithms. The scheme relies on the decisional composite residuosity assumption (DCRA). Let P, Q be Sophie-Germain primes – i.e., $P = 2p + 1$ and $Q = 2q + 1$ for primes p, q. Let $N = PQ$. The DCRA states, roughly, that it is hard to distinguish random elements of $\mathbb{Z}^*_{N^2}$ from random elements of the subgroup consisting of all N-th powers of elements in $\mathbb{Z}^*_{N^2}$. The original DCRA introduced by [22] does not require the use of Sophie-Germain primes, though they are required by [6] and by us for technical reasons. As pointed by [6], it is clear that as long as Sophie-Germain primes are sufficiently dense in the set of primes (as is believed to be true), then the DCRA without the Sophie-Germain restriction implies the DCRA with the Sophie-Germain restriction.

We give a sketch of the encryption scheme; details can be found in [6]. The user creates a composite modulus $N = PQ$ as above. The user's public key includes a collision-resistant hash function H, $h = 1 + N$, a random $g' \in (\mathbb{Z}/N^2\mathbb{Z})^*$, and values $g = g'^{2N}$, $y_1 = g^{x_1}$, $y_2 = g^{x_2}$, and $y_3 = g^{x_3}$ for secrets $x_1, x_2, x_3 \in [N^2/4]$.

To encrypt r with a "label" $L \in \{0,1\}^*$, the sender chooses $t \in_R [N/4]$ and computes (u, e, v) with $u = g^t$, $e = y_1^t h^r$, and $v = \text{abs}((y_2 y_3^{H(u,e,L)})^t)$, where $\text{abs}(a) = N^2 - a \bmod N^2$ if $a > N^2/2$ else $\text{abs}(a) = a \bmod N^2$. The ciphertext is (u, e, v, L). A user with the private key can decrypt (u, e, v, L) as follows. First, it checks that $\text{abs}(v) = v$ and $u^{2(x_2 + H(u,e,L)x_3)} = v^2$. If the check fails, the user outputs $\perp$ and halts. Otherwise, it computes $\hat{r} = (e/u^{x_1})^{2k}$ for $k = 2^{-1} \bmod N$. If $\hat{r}$ is of the form h^r for some $r \in [N]$, it outputs r; otherwise, it outputs $\perp$.

To obtain a verifiable encryption scheme from the basic encryption scheme above, one uses an additional composite modulus $N_2 = P_2 Q_2$, where $P_2 = 2p_2 + 1$ and $Q_2 = 2q_2 + 1$ are safe primes, along with elements $g_2, h_2 \in \mathbb{Z}^*_{N_2}$ of order $p_2 q_2$. Optionally, one may use a third group – e.g., a group Γ of prime order ρ with generators γ and δ for which the discrete logarithm problem is not known to be vulnerable to subexponential attacks – to improve efficiency. We view $(N_2, g_2, h_2, \Gamma, \gamma, \delta)$ as a common reference string. We require $N_2 \neq N$ and $|\Gamma| < N 2^{-k-k'-3}$ for security parameters k and k' as described in [6].

Now, suppose that $\alpha = \gamma^r$ for $r \in [\rho]$; then, r can be verifiably encrypted as follows. The sender computes (u, e, v) as before, generates $s \in_R [N_2/4]$, sets

$\ell = g_2^m h_2^s$, and then provides the following ZKPoK:

$$PK\{(t,r,s) : u^2 = g^{2t} \wedge e^2 = y_1^{2t} h^{2r} \wedge v^2 = (y_2 y_3^{H(u,e,L)})^{2t} \quad (1)$$
$$\wedge\ \alpha = \gamma^r \wedge \ell = g_2^m h_2^s \wedge r \in [\rho]\} \ . \quad (2)$$

The verifiability aspect of the encryption scheme relies on the strong RSA assumption – namely that, given N_2 and $z \in \mathbb{Z}_{N_2}^*$, it is hard to find $x \in \mathbb{Z}_{N_2}^*$ and $e \geq 2$ such that $x^e = z \bmod N_2$. One could alternatively avoid using the third group Γ by setting requiring $N_2 < N2^{-k-k'-3}$, setting $a = g_2^r$, and prove the same equalities as above except those involving ℓ. Notice, however, that if we do use the third group, then the last proof simply becomes a group membership check.

3 Model and Definition of Designated Confirmer Signatures

We describe designated confirmer signatures (DCS) following the exposition of [17]. The model comprises three parties: a signer $\mathcal{S}$, a verifier $\mathcal{V}$, and a designated confirmer $\mathcal{C}$. A designated confirmer signature scheme supports the following (probabilistic polynomial-time) algorithms:

- DCGen: takes as input 1^λ, and outputs two pairs of keys $(\mathsf{SGk}_\mathcal{S}, \mathsf{VFk}_\mathcal{S})$ and $(\mathsf{Pk}_\mathcal{C}, \mathsf{Sk}_\mathcal{C})$. The first pair constitutes $\mathcal{S}$'s signing and verification keys, and the second consists of $\mathcal{C}$'s public and private keys. For simplicity of exposition we denote DCGen as a single algorithm; in an actual implementation, the signer and confirmer would generate their key pairs separately, using distinct algorithms SGen and CGen, so that $\mathcal{C}$ does not learn $\mathsf{SGk}_\mathcal{S}$ and $\mathcal{S}$ does not learn $\mathsf{Sk}_\mathcal{C}$.
- Sign: takes as input a message m and $\mathsf{SGk}_\mathcal{S}$. It outputs a signature σ such that $\mathsf{Verify}(m, \sigma, \mathsf{VFk}_\mathcal{S}) = \mathsf{Accept}$.
- Verify: takes as input $m, \sigma, \mathsf{VFk}_\mathcal{S}$ and outputs Accept if σ is an output of $\mathsf{Sign}(m, \mathsf{SGk}_\mathcal{S})$.

Further, a DCS scheme must support the following protocols:

- $\mathsf{ConfirmedSign}_{(\mathcal{S},\mathcal{V})}$: an interactive protocol between $\mathcal{S}$ and $\mathcal{V}$ with common input $(m, \mathsf{VFk}_\mathcal{S}, \mathsf{Pk}_\mathcal{C})$. The output is a pair (b, σ') where $b \in \{\mathsf{Accept}, \perp\}$ and σ' is $\mathcal{S}$'s designated confirmer signature. For some $\mathcal{V}$, the $\mathsf{ConfirmedSign}$ protocol must be complete and sound. For completeness, we require that there is some $\mathcal{S}$ such that for any (valid) signer and confirmer keys, and for any message m, the $\mathsf{ConfirmedSign}$ protocol outputs a $(\mathsf{Accept}, \sigma')$ where $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', \mathsf{Sk}_\mathcal{C}, \mathsf{VFk}_\mathcal{S}), \mathsf{VFk}_\mathcal{S}) = \mathsf{Accept}$. For soundness, we require that for all signers $\mathcal{S}'$, if the result of running $\mathsf{ConfirmedSign}$ results in an Accept, then

$$\Pr[\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', \mathsf{Sk}_\mathcal{C}, \mathsf{VFk}_\mathcal{S}), \mathsf{VFk}_\mathcal{S}) = \perp] < \mathsf{negl}(\lambda).$$

  In other words, $\mathcal{S}'$ cannot convince $\mathcal{V}$ that an “un-extractable” designated verifier signature is valid.

- $\mathsf{ReconfirmedSign}_{(\mathcal{S},\mathcal{V})}$: an interactive protocol between $\mathcal{S}$ and $\mathcal{V}$ with common input $(m, \mathsf{VFk}_{\mathcal{S}}, \mathsf{Pk}_{\mathcal{C}}, \sigma')$ for designated confirmer signature σ'. The output is $b \in \{\mathsf{Accept}, \perp\}$. The completeness and soundness requirements are similar to those of $\mathsf{Confirm}_{\mathcal{C},\mathcal{V}}$ below. In our scheme, the $\mathsf{ReconfirmedSign}$ protocol is identical to $\mathsf{ConfirmedSign}$ (except that $\mathsf{ReconfirmedSign}$ takes σ' as input) and to the $\mathsf{Confirm}$ protocol (except that a signer takes the place of the confirmer); so, we omit further discussion of $\mathsf{ReconfirmedSign}$.
- $\mathsf{Extract}$: takes as input $m, \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}}$ and returns a string σ such that $\mathsf{Verify}(m, \sigma, \mathsf{VFk}_{\mathcal{S}})$ outputs Accept if σ is an output of $\mathsf{Sign}(m, \mathsf{SGk}_{\mathcal{S}})$, and outputs $\perp$ otherwise.
- $\mathsf{Confirm}_{(\mathcal{C},\mathcal{V})}$: an interactive protocol between $\mathcal{C}$ and $\mathcal{V}$ with common input $(m, \sigma', \mathsf{VFk}_{\mathcal{S}}, \mathsf{Pk}_{\mathcal{C}})$. The output is $b \in \{\mathsf{Accept}, \perp\}$. The protocol must be both complete and sound. For completeness, we require that there is some $\mathcal{C}$ such that if $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}}), \mathsf{VFk}_{\mathcal{S}}) = \mathsf{Accept}$ then $b = \mathsf{Accept}$. For soundness, we require that for all confirmers $\mathcal{C}'$ if

$$\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}}), \mathsf{VFk}_{\mathcal{S}}) = \perp,$$

  then $\Pr[\mathsf{Confirm}_{(\mathcal{C}',\mathcal{V})}(m, \sigma', \mathsf{VFk}_{\mathcal{S}}, \mathsf{Pk}_{\mathcal{C}}) = \mathsf{Accept}] < \mathsf{negl}(\lambda)$.
- $\mathsf{Disavowal}_{(\mathcal{C},\mathcal{V})}$: an interactive protocol between confirmer $\mathcal{C}$ and verifier $\mathcal{V}$ with common input $(m, \sigma', \mathsf{VFk}_{\mathcal{S}}, \mathsf{Pk}_{\mathcal{C}})$. The output is $b \in \{\mathsf{Accept}, \perp\}$. The protocol must be complete and sound. For completeness, we require that there is a confirmer $\mathcal{C}$ such that if $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}}), \mathsf{VFk}_{\mathcal{S}}) = \perp$ then $\mathsf{Disavowal}_{(\mathcal{C},\mathcal{V})} = \mathsf{Accept}$. For soundness, we require that for all confirmers $\mathcal{C}'$, if $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}}), \mathsf{VFk}_{\mathcal{S}}) = \mathsf{Accept}$, then $\Pr[\mathsf{Disavowal}_{(\mathcal{C}',\mathcal{V})}(m, \sigma', \mathsf{VFk}_{\mathcal{S}}, \mathsf{Pk}_{\mathcal{C}}) = \mathsf{Accept}] < \mathsf{negl}(\lambda)$.

For the purposes of the security model, we also define $\mathsf{OutputDCS}_{(\mathcal{S},\mathcal{V})}$, a two-move stunted version of $\mathsf{ConfirmedSign}_{(\mathcal{S},\mathcal{V})}$ in which $\mathcal{V}$ queries m and $\mathcal{S}$ outputs a DCS σ' on m (without "confirming" its correctness).

We now state the security requirements of a DCS scheme in detail.

Definition 1. *Below, we assume that the adversary has access to a collection* $\mathcal{O} = \{\mathsf{ConfirmedSign}_{(\mathcal{S},\mathcal{A})}, \mathsf{ReconfirmedSign}_{(\mathcal{S},\mathcal{A})}, \mathsf{Confirm}_{(\mathcal{C},\mathcal{A})}, \mathsf{Disavow}_{(\mathcal{C},\mathcal{A})}, \mathsf{Extract}_{(\mathcal{C},\mathcal{A})}\}$ *of five oracles for: 1) receiving a confirmed signature on an message of its choice (via the* $\mathsf{ConfirmedSign}_{(\mathcal{S},\mathcal{A})}$ *oracle); 2) executing the prover's role in the* $\mathsf{ReconfirmedSign}_{(\mathcal{S},\mathcal{A})}$ *interactive protocol; 3) executing the prover's role in the* $\mathsf{Confirm}_{(\mathcal{C},\mathcal{A})}$ *interactive protocol; 4) executing the prover's role in the* $\mathsf{Disavow}_{(\mathcal{C},\mathcal{A})}$ *interactive protocol; and 5) extracting an ordinary signature from a designated confirmer signature.*

1. ***Security for verifiers.*** *Security for verifiers follows from the soundness requirement above – informally, that an adversary must not be able, even if the adversary compromises the private keys of* $\mathcal{S}$ *and* $\mathcal{C}$*, to create a* (m, σ') *that will be confirmed (either in* $\mathsf{ConfirmedSign}$ *or* $\mathsf{Confirm}$*) even though* $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}}), \mathsf{VFk}_{\mathcal{S}}) = \perp$ *("Case 1"), or that will be disawowed even though* $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}}), \mathsf{VFk}_{\mathcal{S}}) = \mathsf{Accept}$

("Case 2"). Formally, we define the advantage of the adversary $\mathsf{Adv}^{\mathsf{fool}\mathcal{V}}(\mathcal{A}) := \Pr[b_{\mathsf{fool}\mathcal{V}1} = 1 \vee b_{\mathsf{fool}\mathcal{V}2} = 1 \vee b_{\mathsf{fool}\mathcal{V}3} = 1]$*, where* $(b_{\mathsf{fool}\mathcal{V}1}, b_{\mathsf{fool}\mathcal{V}2}, b_{\mathsf{fool}\mathcal{V}3})$ *are defined by the experiment in Figure 2. For compactness, use "Case 1" and "Case 2" to refer to the verification condition that the adversary's output* (m, σ') *must satisfy. We say a scheme is secure for verifiers if* $\mathsf{Adv}^{\mathsf{fool}\mathcal{V}}(\mathcal{A}) < \mathsf{negl}(\lambda)$ *for all probabilistic polynomial time algorithms* $\mathcal{A}$*.*

2. ***Security for signers.*** *Informally, an adversary should be able to create a DCS* (m, σ'_0) *that is extractable or confirmable (either in* ConfirmedSign *or* Confirm*) only if* σ'_0 *is somehow "equivalent" to a DCS* (m, σ'_1) *that it received in response to a* ConfirmedSign *query on* m*. We say* (m, σ'_0) *and* (m, σ'_1) *are equivalent if* $R(m, \sigma'_0, \sigma'_1) = 1$ *for some specified efficiently computable relation* R*. For example, if DCS signatures are* strongly *existentially unforgeable, it may be appropriate to say* $R(m, \sigma'_0, \sigma'_1) = 1$ *only when* $\sigma'_0 = \sigma'_1$*. However,* R *need not be that restrictive; it depends on the DCS scheme. In the experiment in Figure 2,* L_{sig} *is a list that is viewed as containing the* (m, σ'_1) *associated to the* ConfirmedSign *output, as well as all* (m, σ') *for which* $R(m, \sigma', \sigma'_1) = 1$*. In the figure, for compactness, we say (e.g.)* $\sigma' \notin L_{sig}$ *rather than the more accurate* $(m, \sigma') \notin L_{sig}$*, since* m *will be clear from the context. Formally, we define* $\mathcal{A}$*'s advantage* $\mathsf{Adv}^{\mathsf{imp}\mathcal{S}}(\mathcal{A})$ *to be the probability that the experiment returns 1. We say a scheme is secure for signers if* $\mathsf{Adv}^{\mathsf{imp}\mathcal{S}}(\mathcal{A}) < \mathsf{negl}(\lambda)$ *for all probabilistic polynomial time algorithms* $\mathcal{A}$*.*

3. ***Transcript Simulatability.*** *The confirmation or disavowal of a designated confirmer signature* σ' *should not be* transferable *– e.g., the transcript of a proof of knowledge in* $\mathsf{Confirm}_{\mathcal{C},\mathcal{V}_1}(m, \sigma', \mathsf{VFk}_{\mathcal{S}}, \mathsf{Pk}_{\mathcal{C}})$ *should not convince* $\mathcal{V}_2$ $(\neq \mathcal{V}_1)$ *that* σ' *signs* m*. To ensure that* $\mathcal{V}_1$*'s transcript is unconvincing, we require that transcripts be* simulatable*. To model this in the experiment in Figure 2, first* $\mathcal{A}_0$ *outputs two messages* m_0 *and* m_1 *and some state* s*, next a DCS* σ' *on one of the messages is output, and then* $\mathcal{A}_1$*,* $\mathcal{A}'_1$ *and* $\mathcal{A}_2$ *play a game in which* $\mathcal{A}'_1$ *tries to make its output (when the DCS signs* m_1*) look indistinguishable from* $\mathcal{A}_1$*'s output (when the DCS signs* m_0*);* $\mathcal{A}_2$ *attempts to distinguish whether its input* τ *came from* $\mathcal{A}_1$ *or* $\mathcal{A}'_1$*. In the game,* $\mathcal{A}_1$ *gets almost complete access to the oracles* $\mathcal{O}$*; the only restriction is that* $(m, \sigma') \notin L_{ext}$*, where* L_{ext} *is a list that is viewed as containing each* (m_i, σ'_i) *that has been queried by* $\mathcal{A}_1$ *to the* Extract *oracle, as well all* (m'_i, σ''_i) *for which* $R(m, \sigma'_i, \sigma''_i) = 1$*; otherwise* $\mathcal{A}_1$ *could trivially give* $\mathcal{A}_2$ *indisputable proof that* m_0 *was signed – the extraction of* σ'*. On the other hand, we give* $\mathcal{A}'_1$ *very limited access to* $\mathcal{O}$*; it can make only* q OutputDCS *queries, where* $\mathcal{A}_1$ *makes at most* q ConfirmedSign *queries. We give* $\mathcal{A}_2$ *access to a limited set of oracles* $\mathcal{O}_{lim}$ *– specifically,* $\mathcal{A}_2$ *cannot make any* $\mathcal{O}$ *query on* (m_0, σ'') *if* $R(m_0, \sigma', \sigma'') = 1$ *or on* (m_1, σ'') *if* $R(m_1, \sigma', \sigma'') = 1$*; otherwise, its distinguishing task becomes trivial. If* $\mathcal{A}_2$ *has negligible advantage, this suggests that* $\mathcal{A}_1$*'s potentially authentic transcript that* m_0 *was signed is no more convincing or informative than* $\mathcal{A}'_1$*'s simulated transcript that (falsely) "proves" that* m_0 *was signed. In the security proof,* $\mathcal{A}'_1$ *will use* $\mathcal{A}_1$ *as a subroutine, and will simulate correct responses to* $\mathcal{A}_1$*'s* $\mathcal{O}$ *queries on* σ' *and equivalent*

DCS's. Formally, we define the advantage of the adversary $\mathsf{Adv}^{\mathsf{trans}}(\mathcal{A})$ *to be* $\max\{0, \Pr[\text{experiment returns 1}] - 1/2\}$.

```
Exp-NoFoolVerifier:
1. $(\mathsf{SGk}_{\mathcal{S}}, \mathsf{VFk}_{\mathcal{S}}, \mathsf{Sk}_{\mathcal{C}}, \mathsf{Pk}_{\mathcal{C}}) \leftarrow \mathsf{DCGen}(1^\lambda)$
2. $(m, \sigma'_1, \tau_1, \tau_2, \tau_3) \leftarrow \mathcal{A}_0^{\mathcal{O}}(\pi, \mathsf{SGk}_{\mathcal{S}}, \mathsf{Sk}_{\mathcal{C}})$
3. $(b_{\mathsf{fool}\mathcal{V}1}, \sigma'_0) \leftarrow \mathsf{ConfirmedSign}_{(\mathcal{A}_1(\tau_1), \mathcal{V})}(\pi, m)$ in Case 1
4. $b_{\mathsf{fool}\mathcal{V}2} \leftarrow \mathsf{Confirm}_{(\mathcal{A}_2(\tau_2), \mathcal{V})}(\pi, m, \sigma'_1)$ in Case 1
5. $b_{\mathsf{fool}\mathcal{V}3} \leftarrow \mathsf{Disavowal}_{(\mathcal{A}_3(\tau_3), \mathcal{V})}(\pi, m, \sigma'_1)$ in Case 2
6. Return $b_{\mathsf{fool}\mathcal{V}1} \vee b_{\mathsf{fool}\mathcal{V}2} \vee b_{\mathsf{fool}\mathcal{V}3}$.
```

```
Exp-NoImpSigner:
1. $(\mathsf{SGk}_{\mathcal{S}}, \mathsf{VFk}_{\mathcal{S}}, \mathsf{Sk}_{\mathcal{C}}, \mathsf{Pk}_{\mathcal{C}}) \leftarrow \mathsf{DCGen}(1^\lambda)$
2. $(m, \sigma') \leftarrow \mathcal{A}^{\mathcal{O}}(\pi, \mathsf{Sk}_{\mathcal{C}})$
3. $b_{\mathsf{imp}\mathcal{S}} \leftarrow \mathsf{Verify}(m, \sigma, \mathsf{VFk}_{\mathcal{S}})$
   for $\sigma = \mathsf{Extract}(m, \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}})$
4. Return $(b_{\mathsf{imp}\mathcal{S}} \wedge (\sigma' \notin L_{sig}))$.
```

```
Exp-TranscriptSimulatability:
1. $(\mathsf{SGk}_{\mathcal{S}}, \mathsf{VFk}_{\mathcal{S}}, \mathsf{Sk}_{\mathcal{C}}, \mathsf{Pk}_{\mathcal{C}}) \leftarrow \mathsf{DCGen}(1^\lambda)$
2. $(m_0, m_1, s) \leftarrow \mathcal{A}_0^{\mathcal{O}}(\pi, \mathsf{SGk}_{\mathcal{S}})$
3. $b \xleftarrow{R} \{0, 1\}$
4. $(\mathfrak{b}, \sigma') \leftarrow \mathsf{ConfirmedSign}_{(\mathcal{S}, \mathcal{V})}(\pi, m_b)$
5. If $b = 0$, $\tau \leftarrow \mathcal{A}_1^{\mathcal{O}}(b, m_0, m_1, s, \sigma')$;
   else, $\tau \leftarrow \mathcal{A}_1'^{\,\mathsf{OutputDCS}}(b, m_0, m_1, s, \sigma')$
6. Return 1 iff $b = \mathcal{A}_2^{\mathcal{O}_{lim}}(m_0, m_1, \tau, \sigma')$
   and $\sigma' \notin L_{ext}$.
```

Fig. 2. Experiments for definition of DCS security. Above, π is shorthand for $(1^\lambda, \mathsf{VFk}_{\mathcal{S}}, \mathsf{Pk}_{\mathcal{C}})$.

In our model, we allow that σ' may convince verifier $\mathcal{V}_2$ above that the signer indeed signed *some* message m. In this sense, the transcript is not perfectly simulatable; only the ZK proofs are. Accordingly, in the security model, $\mathcal{A}'_1$ needs access to ConfirmedSign; without *some* DCS's generated by $\mathcal{S}$, $\mathcal{A}'_1$ has no hope in our scheme of making its output indistinguishable from $\mathcal{A}_1$ (which has almost unrestricted access to $\mathcal{O}$). Thus, our model is weaker than that in [21]. However, we believe that our model, especially given our very efficient instantiation, is suitable for real-world settings, where it would be easy (e.g.) for the confirmer to publish a few "dummy" signatures by the signer during each time period to camouflage the presence or absence of a "real" (meaningful) signature. Again, we are inspired here by the discussion of Goldwasser and Waisbard [17], which emphasizes capturing only the "non-verifiability" of a DCS, although our definition differs from theirs.

How the Model Prevents Confirmer Impersonation. Of course, for a DCS scheme to be secure, it should be infeasible for an adversary $\mathcal{A}$ (even if it has $\mathsf{SGk}_{\mathcal{S}}$) to *impersonate the confirmer* by performing an Extract, Confirm, Disavowal, or ConfirmedSign associated to a pair (m, σ') contained in $L_{sig} \setminus L_{ext}$. Interestingly, this requirement is already covered by a combination of our

Exp-NoFoolVerifier and Exp-TranscriptSimulatability experiments. For example, suppose that there is an adversary $(\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2)$ that "breaks" Confirm in that $(m'_0, s') \leftarrow \mathcal{B}_0^{\mathcal{O}}(\pi, \mathsf{SGk}_{\mathcal{S}})$, $(\sigma'', \tau') \leftarrow \mathcal{B}_1^{\mathcal{O}}(\pi, s')$, $\mathsf{Confirm}_{(\mathcal{B}_2(\tau'), \mathcal{V})}(\pi, m'_0, \sigma'') = \mathsf{Accept}$ for $(m'_0, \sigma'') \in L_{sig} \setminus L_{ext}$ with non-negligible probability, where (s', τ') is state information that is forwarded, and where B_1 makes only a polynomial number q of $\mathcal{O}$ queries. Then, $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ can use $(\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2)$ to break Exp-TranscriptSimulatability with non-negligible probability, as follows. $\mathcal{A}_0$ runs $\mathcal{B}_0$, relays $\mathcal{B}_0$'s $\mathcal{O}$ queries and the responses, sets $m_0 = m'_0$ and outputs (m_0, m_1). If $b = 0$ in Exp-TranscriptSimulatability, $\mathcal{A}_1$ runs $\mathcal{B}_1$ and relays $\mathcal{B}_1$'s $\mathcal{O}$ queries and the responses, except that it responds to one of $\mathcal{B}_1$'s ConfirmedSign queries on m_0 by using a ReconfirmedSign query on (m_0, σ'). $\mathcal{B}_1$ outputs (σ'', τ') and $\mathcal{A}_1$ sets $\tau = \tau'$; σ'' is equivalent to σ' with probability at least $1/q$. Finally, if σ'' is equivalent to σ', $\mathcal{A}_2$ runs $\mathsf{Confirm}_{(\mathcal{B}_2(\tau), \mathcal{V})}(\pi, m_0, \sigma'')$. If the output is Accept, $\mathcal{A}_2$ outputs '0'; otherwise, it outputs a random bit. Since the output is Accept with non-negligible probability, and since an output of Accept implies $\mathsf{Verify}(m', \mathsf{Extract}(m', \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}}), \mathsf{VFk}_{\mathcal{S}}) = \mathsf{Accept}$ with overwhelming probability assuming $\mathsf{Adv}^{\mathsf{fool}\mathcal{V}}(\mathcal{A}) < \mathsf{negl}(\lambda)$, $\mathcal{A}_2$'s advantage is non-negligible.

Using the approach above, one can show that, for any (m'_0, m'_1) adaptively chosen by $\mathcal{B}_0$, $\mathcal{B}_1$ has a negligible probability of outputing a DCS σ'' on m'_0 that $\mathcal{B}_2$ can disavow on m'_1. However, for the special case of Disavowal, one may want to require something stronger: that for any m'_0 adaptively chosen by $\mathcal{B}_0$, $\mathcal{B}_1$ has a negligible probability of outputing an m'_1 and a DCS σ'' on m'_0 that $\mathcal{B}_2$ can disavow on m'_1. *Our scheme satisfies this requirement*, but it may not be necessary in general. If the message space is super-polynomial, and if a verifier $\mathcal{V}$ could merely check that m'_1 was generate randomly and independently of (m'_0, σ''), it would already believe that the probability that σ'' is a DCS on m'_1 is negligible. Thus, it does not seem unreasonable to allow that $(\mathcal{B}_1, \mathcal{B}_2)$ may be able to disavow *some* message with respect to a DCS σ'' in $L_{sig} \setminus L_{ext}$ on m'_0. It is an open question whether a more efficient DCS exists that meets the weaker requirement.

4 Our Transformation

THE GENERIC CONSTRUCTION. We first describe a generic scheme that transforms any traditional signature scheme into a DCS scheme; the transformation also requires an IND-CCA2 secure encryption scheme and a statistically-hiding computationally-binding commitment scheme C. The scheme also uses zero knowledge proofs secure against cheating verifiers, as discussed in Section 2. After describing our scheme generically, we provide an efficient instantiation.

– DCGen: $\mathcal{S}$ uses a secure digital signature scheme $\mathsf{DSS} = (\mathsf{SGen}, \mathsf{Sign}, \mathsf{Verify})$, and creates a key pair $(\mathsf{VFk}_{\mathcal{S}}, \mathsf{SGk}_{\mathcal{S}}) \leftarrow \mathsf{SGen}(1^\lambda)$. $\mathcal{C}$ uses an IND-CCA-2 encryption scheme $\mathsf{PKE} = (\mathsf{CGen}, \mathsf{Enc}, \mathsf{Dec})$, and creates key pair $(\mathsf{Pk}_{\mathcal{C}}, \mathsf{Sk}_{\mathcal{C}}) \leftarrow \mathsf{Gen}(1^\lambda)$. Note that $\mathcal{C}$ need not participate in any setup other than creating and publishing a key pair.

- Sign: To sign a message m with auxiliary information c, $\mathcal{S}$ creates a statistically hiding and computationally binding commitment $\psi = C(m, r)$ to the message m using randomness r and creates $\sigma^* = \mathsf{Sign}((\psi, c, \mathsf{VFk}_{\mathcal{S}}), \mathsf{SGk}_{\mathcal{S}})$. The basic signature is $\sigma = (\sigma^*, c, r)$. $\mathcal{S}$'s verification key $\mathsf{VFk}_{\mathcal{S}}$ is signed together with the commitment to prevent a signature issued by one signer from being fed to the $\mathsf{Confirm}$ oracle with a different signer's public key.
- $\mathsf{ConfirmedSign}$: In addition to the above steps in the Sign procedure, $\mathcal{S}$ also computes the ciphertext $c = \mathsf{Enc}(\mathsf{Pk}_{\mathcal{C}}, r)$. The designated confirmer signature is $\sigma' = (\sigma^*, \psi, c)$ for $\sigma^* = \mathsf{Sign}((\psi, c, \mathsf{VFk}_{\mathcal{S}}), \mathsf{SGk}_{\mathcal{S}})$. The signer also performs a ZK proof of knowledge of a value r such that $\psi = C(m, r)$ and $c = \mathsf{Enc}(\mathsf{Pk}_{\mathcal{C}}, r)$.
- $\mathsf{Confirm}$: $\mathcal{C}$ first checks that $(\psi, c, \mathsf{VFk}_{\mathcal{S}})$ has been signed with $\mathsf{SGk}_{\mathcal{S}}$ using the provided $\mathsf{VFk}_{\mathcal{S}}$, and aborts if the check fails. Then, $\mathcal{C}$ performs a ZK proof of knowledge of a value r such $\psi = C(m, r)$.
- $\mathsf{Disavow}$: To disavow a purported signature $\sigma' = (\sigma^*, \psi, c)$ on message m, $\mathcal{C}$ does the following. $\mathcal{C}$ first checks if c is a valid encryption of some r. If not, it performs a ZK proof of knowledge that the string is not a well-formed encryption. Otherwise, $\mathcal{C}$ computes $r' = \mathsf{Dec}(\mathsf{Sk}_{\mathcal{C}}, c)$. If $\psi \neq C(m, r')$, then $\mathcal{C}$ provides a ZKPoK of a value ρ such that $\psi \neq C(m, \rho)$ and $\rho = \mathsf{Dec}(\mathsf{Sk}_{\mathcal{C}}, c)$.
- $\mathsf{Extract}$: On input $\sigma' = (\sigma^*, \psi, c)$ and m, $\mathcal{C}$ computes $r' = \mathsf{Dec}(\mathsf{Sk}_{\mathcal{C}}, c)$ and confirms that $\psi = C(m, r')$ and $\sigma^* = \mathsf{Sign}((\psi, c, \mathsf{VFk}_{\mathcal{S}}), \mathsf{SGk}_{\mathcal{S}})$. If so, it outputs r'; else, it outputs $\perp$.

Notice that all the statements involving zero-knowledge proofs can be expressed as NP statements (and have a short witness). Therefore, we can, in theory, instantiate the above scheme in polynomial time for any suitably secure encryption scheme, commitment scheme, and signature scheme. Of course using generic zero-knowledge proofs for NP-statements is not very practical. Therefore, we now describe how to instantiate the encryption and commitment schemes so that the resulting zero-knowledge proofs of knowledge are simple and efficient.

Efficient Instantiation for Any Signature Scheme. We show how to efficiently instantiate the above scheme. The underlying encryption scheme PKE is the scheme by Camenisch and Shoup discussed in Section 2. The commitment scheme $C(m, r)$ will be a Pedersen-type commitment scheme over group Γ of prime order ρ, with generators γ and δ, as described in Section 2. We can use any secure signature scheme. Then, our confirm and disavow protocols use the CDM ZK proofs as described in Section 2. With these choices, the underlying zero-knowledge proofs are efficient (using the CDM techniques for proving equality and inequality of discrete logarithms together with the Camenisch-Shoup verifiable encryption of the randomness used for the Pedersen commitment). Moreover, we can plug in *any* secure signature scheme and the complexity of the $\mathsf{Confirm}$, $\mathsf{Disavow}$, and $\mathsf{Extract}$ are essentially independent of this choice.

Security Analysis. We now state a theorem that our generic transformation yields a secure designated confirmer signature scheme. We require that the signature schemes be existentially unforgeable under chosen message attack, that our commitment schemes be hiding and binding, and the encryption scheme be secure against chosen ciphertext attack. The security for our efficient Paillier-based instantiation follows.

Theorem 1. *Let* $\mathsf{DSS} = (\mathsf{SGen}, \mathsf{Sign}, \mathsf{Verify})$ *be any signature scheme existentially unforgeable against chosen message attack, and let* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be any IND-CCA2 secure encryption scheme and* $C(M, r)$ *be any statistically-hiding computationally-binding commitment scheme with perfect zero-knowledge proofs of knowledge for committed values secure against cheating verifiers. Then the DCS scheme obtained by our generic conversion is a secure DCS scheme.*

Proof. (Sketch). In this proof, we say that the relation $R(m, \sigma', \sigma'')$ equals 1 in our scheme if $\sigma' = (\sigma^*, \psi, c)$, $\sigma'' = (\sigma^{*\prime}, \psi', c')$, and $(\psi, c) = (\psi', c')$; otherwise, it equals 0. Also, below, a second algorithm (say, $\mathcal{B}$) will construct the necessary zero knowledge proofs in response to to $\mathcal{A}$'s $\mathcal{O}$ queries by using a (possibly rewinding) simulator. For example, assuming we use the CDM protocol [13] for our ZK proofs, $\mathcal{B}$ proceeds as follows: upon receiving $\mathcal{A}$'s commitment to its value e together with a proof of knowledge of e, $\mathcal{B}$ extracts e by using the knowledge extractor E together with $\mathcal{A}$. Then $\mathcal{B}$ can complete Part 2 of the CDM protocol by using its knowledge of e. As Cramer et al. argue, B's proof in Part 2 is witness indistinguishable; so, $\mathcal{B}$'s simulation is sound.

Now, suppose the theorem is false. Then there exists a probabilistic polynomial time adversary $\mathcal{A}$ that can break the security of the DCS scheme. Specifically, at least one of $\mathsf{Adv}^{\mathsf{imp}\mathcal{S}}(\mathcal{A}), \mathsf{Adv}^{\mathsf{imp}\mathcal{C}}(\mathcal{A})$, $\mathsf{Adv}^{\mathsf{fool}\mathcal{V}}(\mathcal{A})$, or $\mathsf{Adv}^{\mathsf{trans}}(\mathcal{A})$ is not negligible in the security parameter. We consider the resulting cases.

The $\mathsf{Adv}^{\mathsf{imp}\mathcal{S}}(\mathcal{A})$ Case. If $\mathsf{Adv}^{\mathsf{imp}\mathcal{S}}(\mathcal{A})$ is not negligible, then the adversary has a non-negligible probability of successfully outputting a DCS (m, σ') for which $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', \mathsf{Sk}_{\mathcal{C}}, \mathsf{VFk}_{\mathcal{S}}), \mathsf{VFk}_{\mathcal{S}}) = \mathsf{Accept}$ and $\sigma' \notin L_{sig}$. From $\mathcal{A}$, we can construct an algorithm $\mathcal{B}$ that either constructs an existential forgery of the underlying signature scheme, or violates the binding property of the commitment scheme as follows.

The algorithm $\mathcal{B}$ generates $(\mathsf{Sk}_{\mathcal{C}}, \mathsf{Pk}_{\mathcal{C}})$ and gives $(\pi, \mathsf{Sk}_{\mathcal{C}})$ to $\mathcal{A}$. Then $\mathcal{B}$ responds to $\mathcal{A}$'s $\mathsf{ConfirmedSign}$ query on a message m' by choosing a random r', setting $\psi = C(m', r')$, generating an appropriate ciphertext c that encrypts r', and then using its oracle access to Sign to obtain a signature on (ψ, c). For $\mathsf{Confirm}$, $\mathsf{Extract}$ and $\mathsf{Disavowal}$ queries, $\mathcal{B}$ uses $\mathsf{Sk}_{\mathcal{C}}$. Suppose that $\mathcal{A}$ outputs a pair (m, σ') with $\sigma' = (\sigma^*, \psi,)$ with $(m, \sigma') \notin L_{sig}$. $\mathcal{B}$ uses $\mathsf{Sk}_{\mathcal{C}}$ to perform $\mathsf{Extract}$ on σ', thereby obtaining $\sigma = (\sigma^*, r, c)$. If $(m', \sigma') \in L_{sig}$ for some m', then $\mathcal{B}$ must have responded to $\mathcal{A}$'s $\mathsf{ConfirmedSign}$ query on m' by generating a random r' for which $\psi = C(m', r') = C(m, r)$; since $m' \neq m$, this violates the binding property of the commitment scheme. Otherwise, if there is no message m' for which $(m', \sigma') \in L_{sig}$, $\mathcal{B}$ outputs σ as an existential forgery of the signature scheme (on the message $(\psi, c, \mathsf{VFk}_{\mathcal{S}})$).

The $\mathsf{Adv}^{\mathsf{imp}\mathcal{C}}(\mathcal{A})$ Case. Suppose that $\mathsf{Adv}^{\mathsf{imp}\mathcal{C}}(\mathcal{A})$ is not negligible. Then, from $\mathcal{A}$, we can construct an algorithm $\mathcal{B}$ that breaks the chosen-ciphertext security of the underlying encryption scheme.

The algorithm $\mathcal{B}$ runs as follows. It picks two random messages r_0 and r_1 for the "find" stage of the encryption game, and receives a challenge ciphertext $\mathsf{Enc}(r_b)$ for a random b. Then $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine. $\mathcal{B}$ responds to $\mathcal{A}$'s $\mathsf{Extract}$ queries by using its access to the decryption oracle of the encryption scheme.

Then $\mathcal{B}$ responds to *one* of $\mathcal{A}$'s q $\mathsf{ConfirmedSign}$ queries by flipping a coin b', setting $\psi = C(m, r_{b'})$ and $c = \mathsf{Enc}(r_b)$, querying the Sign oracle for a signature σ^* on $(\psi, c, \mathsf{VFk}_{\mathcal{S}})$, and setting $\sigma' = (\sigma^*, \psi, c)$. To construct a (potentially false) "proof" that the $r_{b'}$ embedded in ψ is identical to the r_b embedded in c, $\mathcal{B}$ uses the simulator $S^{\mathcal{A}}$ for the ZK proof of knowledge of r_b, treating the adversary as a cheating verifier. Because the ZK proof of knowledge is complete and because the transcripts output by the simulator are identically distributed to the interaction between the real prover and the adversary, the adversary will accept the "proof."

For *other* $\mathsf{ConfirmedSign}$ queries by $\mathcal{A}$, $\mathcal{B}$ acts as follows: $\mathcal{B}$ generates a random r as in the actual scheme, sets $\psi = C(m, r)$ and $c = \mathsf{Enc}(r)$, and queries the Sign oracle for a signature σ^* on $(\psi, c, \mathsf{VFk}_{\mathcal{S}})$; it responds with (σ^*, ψ, c) and the appropriate proofs of knowledge.

$\mathcal{B}$ responds to a $\mathsf{Confirm}$ or $\mathsf{Disavowal}$ query on $(\sigma^*, \psi, c, \mathsf{VFk}_{\mathcal{S}}, m)$ by determining whether σ^* is a valid signature on $(\psi, c, \mathsf{VFk}_{\mathcal{S}})$.

If so, and if $\mathcal{B}$ has not queried $(\psi, c, \mathsf{VFk}_{\mathcal{S}})$ to the Sign oracle, $\mathcal{B}$ aborts. If $(\psi, c, \mathsf{VFk}_{\mathcal{S}})$ has already been signed, then $\mathcal{B}$ recovers its log of the action it performed in the $\mathsf{ConfirmedSign}$ query corresponding to $(\psi, c, \mathsf{VFk}_{\mathcal{S}})$; in particular, it recovers the values of (r', m') that it used to generate ψ. Then, if $m = m'$ (and $\mathsf{Confirm}$ is therefore appropriate), it proves (using a "false" proof via the simulator, if necessary) that this value of r' is encrypted in c. Analogously, if $m \neq m'$ (and $\mathsf{Disavowal}$ is therefore appropriate), $\mathcal{B}$ can provide a proof of knowledge of an r' such that r' is encrypted in c and $C(m, r') \neq \psi$.

Eventually, with non-negligible probability, $\mathcal{A}$ outputs some $(m, \sigma') \in L_{sig} \setminus L_{ext}$. Let $\sigma' = (\sigma^*, \psi, c)$. If $\mathcal{A}$ performs a successful $\mathsf{Confirm}$, $\mathsf{Disavowal}$ or $\mathsf{ConfirmedSign}$ using (m, σ'), $\mathcal{B}$ uses $\mathcal{A}$ together with the knowledge extractor E to extract the value r encrypted in c. Now there are two cases. In the first case, $c = \mathsf{Enc}(r_b)$ – i.e., the challenge ciphertext for $\mathcal{B}$ – with probability negligibly close to $1/q$. Notice that because the zero-knowledge proof is perfect zero-knowledge, the use of the simulator to construct proofs does not affect this probability. If $c = \mathsf{Enc} r_b$, then $\mathcal{B}$ outputs b as its guess; otherwise, $\mathcal{B}$ selects b uniformly at random. In the second case, $c \neq \mathsf{Enc}(r_b)$. In this case, we can violate the soundness of the underlying ZKPoK: the execution of $\mathsf{Confirm}$ or $\mathsf{ConfirmedSign}$ on c constitutes an interaction that violates soundness of the proof of knowledge.

THE $\mathsf{Adv}^{\mathsf{trans}}$ CASE. We construct $\mathcal{A}_1'$ by using $\mathcal{A}_1$ as a subroutine; later, we show that if $\mathcal{A}_2$ can distinguish which output came from $\mathcal{A}_1$, this violates the IND-CCA2 security of the encryption scheme.

We construct $\mathcal{A}_1'$ as follows. To generate τ, $\mathcal{A}_1'$ runs $\mathcal{A}_1$ on input (m_1, s, σ'). $\mathcal{A}_1'$ responds to $\mathcal{A}_1$'s permitted $\mathcal{O}$ queries by using its access to $\mathcal{O}$. For $\mathsf{Confirm}$ queries on (m_1, σ') (or on (m_1, σ_1') for which $R(m_1, \sigma', \sigma_1') = 1$)and disavowal queries on (m_0, σ') (or equivalent DCS's), $\mathcal{A}_1'$ uses the rewinding technique to construct the needed (false) ZK proofs. Eventually, $\mathcal{A}_1$ outputs a string τ'; $\mathcal{A}_1'$ outputs $\tau = \tau'$ and terminates. Let $1/2 + \epsilon_0$ be the probability that $\mathcal{A}_2$ outputs b when $\mathcal{A}_1'$ generates its output in this manner.

Now, consider the following modified experiment ModExp, whose only difference from the experiment in Figure 2 is that we replace $\mathcal{A}_1$ with $\mathcal{A}_1''$. We construct $\mathcal{A}_1''$ as follows. To generate τ, $\mathcal{A}_1''$ runs $\mathcal{A}_1$ on input (m_0, s, σ'). $\mathcal{A}_1''$ responds to $\mathcal{A}_1$'s permitted $\mathcal{O}$ queries by using its access to $\mathcal{O}$. For $\mathsf{Confirm}$ queries on (m_0, σ') (or equivalent DCS's) and disavowal queries on (m_1, σ') (or equivalent DCS's), $\mathcal{A}_1''$ uses the simulator to construct the needed ZK proofs. Eventually, $\mathcal{A}_1$ outputs a string τ'; $\mathcal{A}_1''$ outputs $\tau = \tau'$ and terminates. Let $1/2 + \epsilon_1$ be the probability that $\mathcal{A}_2$ outputs b when $\mathcal{A}_1''$ generates its output in this manner.

The only difference between the two experiments is $\mathcal{A}_1$'s view; in ModExp, $\mathcal{A}_1$ obtains simulated proofs of true statements in response to its $\mathsf{Confirm}$ query on (m_0, σ') and disavowal query on (m_1, σ'). Thus, if $|\epsilon_0 - \epsilon_1|$ is non-negligible, then $\mathcal{A}_2$ distinguishes interactions with the simulator from interactions with the true prover, contradicting the zero-knolwedge property of the ZK proofs.

The only difference between the algorithms $\mathcal{A}_1'$ and $\mathcal{A}_1''$ in ModExp is that the former simulates a (false) $\mathsf{Confirm}$ on (m_1, σ'), while the latter simulates a true $\mathsf{Confirm}$ (and similarly for $\mathsf{Disavowal}$ queries). Thus, if $|\epsilon_1|$ is non-negligible, an adversary $\mathcal{B}$ can use $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_1', \mathcal{A}_1'')$ to break the IND-CCA2 security of the encryption scheme. Specifically, $\mathcal{B}$ runs $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_1', \mathcal{A}_1'')$ and obtains the messages m_0 and m_1. Then $\mathcal{B}$ selects m_0 and m_1 in the find stage of the IND-CCA2 encryption game. Finally, $\mathcal{B}$ guesses the bit b. We see that if $|\epsilon_1|$ is non-negligible, then $\mathcal{B}$ wins the encryption game with non-negligible probability.

The $\mathsf{Adv}^{\mathsf{fool}\mathcal{V}}(\mathcal{A})$ Case. Finally, if $\mathsf{Adv}^{\mathsf{fool}\mathcal{V}}(\mathcal{A})$ is not negligible, then the adversary can generate fake valid zero-knowledge proofs with non-negligible probability, violating soundness.

5 Evaluation

Our Efficient Instantiation. For our efficient instantiation, $\mathsf{Confirm}$ requires proving equality of discrete logarithms, specifically proving knowledge of an r such that $\frac{e}{u^{x_1}}^2 = \gamma$. This can be accomplished using protocols of Chaum and Pedersen in four exponentiations. To achieve general-verifier ZK, the techniques of Cramer et al. result in a 4-round protocol with 10 total exponentiations [8, 13].

The $\mathsf{Disavow}$ protocol requires proving inequality of discrete logarithms, which we do by using the techniques of Camenisch and Shoup [6]. From the Preliminaries, the resulting proof consists of five clauses, four of which prove statements about discrete logarithms and the final clause shows that a committed value r is in a specified range. Because we work over a group with public prime order and the range is just the order of the group, the range test reduces to a simple group membership test costing one exponentiation. For the other four clauses, we can apply the optimized protocol of Cramer et al. to obtain general-verifier ZK at the cost of 4 rounds and 10 exponentiations per clause; with sequential composition this gives us 16 total rounds and 41 total exponentiations. We remark that a more efficient protocol appears possible by using the results of Cramer et al. on monotone composition of SHVZK protocols, but this result is already better than a generic zero-knowledge proof [13].

Comparison to Goldwasser-Waisbard. Both our approach and Goldwasser-Waisbard use a weakened definition of designated confirmer signatures which requires non-verifiability of unconfirmed signatures. Goldwasser-Waisbard use this weakening to explore strong witness hiding proofs of knowledge (WHPOKs) for Confirm protocols, and we use this weakening to explore a different way of creating designated confirmer signatures [17].

While the strong WHPOKs constructed by Goldwasser and Waisbard are more efficient than generic zero-knowledge proofs, they still require substantial practical overhead. For example, the strong WHPOK described for the case of Cramer-Shoup signatures uses a zero-knowledge proof of knowledge (ZKPOK) of an i^{th} root as a subroutine. Each such proof of an i^{th} root requires an exponentiation; with the suggested parameters this uses a 161-bit exponent. Two proofs are needed for the WHPOK, which then must be repeated λ times to reduce the soundness error. As a result, Confirm requires 2λ exponentiations. Further, Disavow still requires a generic ZKPOK; Goldwasser and Waisbard note that there appears to be no easy way to extend their approach to obtain an efficient Disavow, since it is not clear what witness is supposed to be "hidden." The efficiency requirements for the strong WHPOK exhibited for the GMR and Gennaro-Halevi-Rabin signatures are similar. While Goldwasser and Waisbard do exhibit a more efficient WHPOK for the case of RSA signatures, the resulting DCS signatures are existentially forgeable.

Our Confirm, in contrast, requires 10 exponentiations. Further, our Disavow protocol is more efficient than the generic ZKPOK used by Goldwasser-Waisbard, although less efficient than Confirm. Finally, our protocols are zero-knowledge instead of witness-hiding.

The main advantage of Goldwasser-Waisbard is that they have exhibited efficient strong WHPOK using the same assumptions as the underlying signature scheme. Our approach, in contrast, requires the "extra" composite residuosity assumption for the Paillier scheme. We note, however, that for each new signature scheme, new effort must be exerted to find efficient strong WHPOK without adding new assumptions. Conversely, one could look for protocols in our framework that require different assumptions for Confirm and Disavow. For example, if one were willing to live with an inefficient Disavow , we could replace the Camenisch-Shoup encryption with an arbitrary IND-CCA2 scheme and construct an efficient Confirm assuming only hardness of discrete logarithms.

Comparison to Camenisch-Michels. Camenisch and Michels give a generic scheme for constructing designated confirmer signatures. They propose a specific instantiation with RSA signatures and Cramer-Shoup encryption [4]. Security for the underlying RSA signature is achieved by using full-domain hash, so the resulting scheme has a proof of security only in the Random Oracle model. Their

Confirm protocol requires proving 12 statements regarding equalities and proofs that committed numbers are in a specific interval, while their Disavow protocol has 20 such statements. They note in their Remark 4 that because these proofs involve double discrete logarithms the verifier uses only binary challenges. As a result, the proof must be repeated λ times for soundness [4]. We optimistically estimate that each clause takes 3 exponentiations, leading to a total of 36λ exponentiations for Confirm and 60λ for Disavow.

Comparison to Camenisch-Shoup. In their paper on verifiable encryption, Camenisch and Shoup observe that, following Asokan et al., a designated confirmer Schnorr signature can be created where Confirm requires proving only a single equality of discrete logarithms [6, 1]. The details are due to appear in a forthcoming paper. Because this paper is not yet available, we speculate on the details to make a reasonable comparison to our work. Let (γ, γ^x) be a public key for a Schnorr signature, where γ is a generator of a group G and x is the secret key. Then a Schnorr signature on m is the triple (β, c, s), where $\beta = \gamma^r$ for a random r, $c = H(\beta, m)$, and $s = r + xc \bmod \rho$, for ρ the order of G. The DCS Schnorr output is then the 4-tuple (β, c, δ, ψ), where $\delta = \gamma^s$ and ψ is an encryption of s with the confirmer's public key. Anyone can check that $\delta = \beta\gamma^c$ to verify the consistency of the signature. Then the confirmer need only prove or disprove that $\psi = E(\log_\gamma \delta)$ to Confirm or Disavow.

In this special case, the Camenisch-Shoup approach is as efficient as our scheme for Confirm and Disavow; indeed, we use their protocols for proving inequality of discrete logarithms. Unfortunately, the Schnorr scheme requires random oracles, so as sketched the approach does not produce a scheme with a proof in the standard model.

If we review the Cramer-Shoup, Goldwasser-Micali-Rivest, and Gennaro-Halevi-Rabin signature schemes with proofs of security in the standard model, then we see that these schemes do not appear to have the same reduction as Schnorr from validity to equality of discrete logarithms. For example, Cramer-Shoup requires proving knowledge of an i'th root, which does not translate straightforwardly to a statement about equality of discrete logarithms. In contrast, our use of a commitment adds a "layer of indirection" that allows us to achieve efficiency for every signature scheme. As a result, we can use any of these signature schemes to obtain an efficient designated confirmer signature with a proof in the standard model.

Finally, we note that the Camenisch-Shoup approach requires, as we do, a Paillier-type encryption and the associated composite residuosity assumption for efficient implementation. Therefore both their approach and ours require possibly introducing extra assumptions beyond those of the underlying signature scheme.

6 Conclusion

We have shown that weakening the definition of designated confirmer signatures, as suggested by Goldwasser and Waisbard, can yield a big payoff in the efficiency of generic designated confirmer signature schemes. By using a commitment scheme to add a "layer of indirection," we used the techniques of Camenisch and Shoup to exhibit efficient Confirm and Disavow protocols for any underlying signature scheme. Going further, we could look for commitment schemes and efficient protocols based on different assumptions. For example, can we adapt the techniques of Camenisch and Lysyanskaya [5] to obtain an even more efficient instantiation based on bilinear mappings? We could also investigate the strong witness hiding proofs of knowledge approach of Goldwasser and Waisbard with an eye towards weakening the assumptions required for efficient instantiation.

7 Acknowledgments

We thank Phil MacKenzie for his careful review and assistance. We thank Jan Camenisch for answering our questions regarding Schnorr designated confirmer signatures. We also thank the anonymous referees for useful comments. This work was carried out while David Molnar was visiting NTT DoCoMo Labs USA.

References

1. N. Asokan and V. Shoup and M. Waidner. "Optimistic fair exchange of digital signatures." IEEE Journal on Selected Areas in Communications 18 (2000) no. 4 591-610.
2. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *ACM CCS 1993.*
3. M. Bellare and O. Goldreich. On defining proofs of knowledge. In CRYPTO '92. Lecture Notes in Computer Science, no. 740 390-420.
4. J. Camenisch and M. Michels. "Confirmer Signature Schemes Secure Against Adaptive Adversaries." *EUROCRYPT 2000.*
5. J. Camenisch and A. Lysyanskaya. "Signature Schemes and Anonymous Credentials from Bilinear Maps." *CRYPTO 2004.*
6. J. Camenisch, V. Shoup. "Practical Verifable Encryption and Decryption of Discrete Logarithms" *CRYPTO 2003.* Full version available from `www.shoup.net`.
7. D. Chaum "Designated Confirmer Signatures," *EUROCRYPT 1994.*
8. D. Chaum and T. P. Pedersen "Wallet Databases with Observers." *CRYPTO 1992.*
9. D. Chaum and H. Van Antwerpen. Undeniable Signatures. *CRYPTO 1989.*
10. R. Cramer and V. Shoup "Signature Schemes Based on the Strong RSA Assumption" *ACM CCS 1999.*
11. I. Damgard. "Efficient concurrent zero-knowledge in the auxiliary string model." *EUROCRYPT 2000.*
12. R. Cramer and V. Shoup. "A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Message Attack." *CRYPTO 1998.*
13. R. Cramer, I. Damgard and V. Shoup. "Efficient Zero-Knowledge Proofs of Knowledge without Intractability Assumptions." *PKC 2000.*
14. R. Gennaro, S. Halevi, and T. Rabin. "Secure Hash-and-Sign Signatures Without the Random Oracle." *Proc. EUROCRYPT 1999.*
15. O. Goldreich and A. Kahan. "How to Construct Constant-Round Zero-Knowledge Proof Systems for NP." *Journal of Cryptology*, 9(3):167–189.

16. S. Goldwasser, S. Micali, and R. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks." *SICOMP Vol 17, No 2, April 1988, pp. 281–308.*
17. S. Goldwasser and E. Waisbard. "Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes," *Theory of Cryptography Conference, 2004.*
18. M. Jakobsson, K. Sako, R. Impagliazzo. "Designated Verifier Proofs and Their Applications." *EUROCRYPT 1996.*
19. R. Cramer, I. Damgard, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *CRYPTO 1994.*
20. M. Naor and M. Yung. "Public-key cryptosystems provably secure against chosen ciphertext attacks." *STOC 1990.*
21. T. Okamoto "Designated Confirmer Signatures and Public Key Encryption Are Equivalent," *CRYPTO 1994.*
22. P. Paillier. "Public-key cryptosystems based on composite degree residue classes." *EUROCRYPT 1999.*
23. T. P. Pedersen. "A threshold cryptosystem without a trusted third party." *EUROCRYPT 1991.*
24. C. P. Schnorr. "Efficient Identification and Signatures for Smart Cards." *CRYPTO 1989.*
25. A. Shamir, Y. Tauman "Improved Online-Offline Signature Schemes," *CRYPTO 2001.*
26. M. Michels and M. Stadler. "Generic constructions for secure and efficient confirmer signature schemes." *EUROCRYPT 1998.*
27. J. Garay and P. MacKenzie and K. Yang "Strengthening Zero-Knowledge Protocols Using Signatures." EUROCRYPT 2003, LNCS 2656, pp. 177–194
28. P. MacKenzie and K. Yang. "On Simulation-Sound Trapdoor Commitments." EUROCRYPT 2004, LNCS 3027, pp. 382–400.