Designated Confirmer Signatures Revisited[⋆]

Douglas Wikström

ETH Zürich, Department of Computer Science
douglas@inf.ethz.ch

Abstract. Previous definitions of designated confirmer signatures in the literature are incomplete, and the proposed security definitions fail to capture key security properties, such as unforgeability against malicious confirmers and non-transferability. We propose new definitions.
Previous schemes rely on the random oracle model or set-up assumptions, or are secure with respect to relaxed security definitions. We construct a practical scheme that is provably secure with respect to our security definition under the strong RSA-assumption, the decision composite residuosity assumption, and the decision Diffie-Hellman assumption.
To achieve our results we introduce several new relaxations of standard notions. We expect these techniques to be useful in the construction and analysis of other efficient cryptographic schemes.

1 Introduction

In a digital signature scheme, as introduced by Diffie and Hellman [10], a signer computes a signature of a message using its secret key, and then anybody holding the public key can verify the signature. This means that the receiver of a signature can show the signature to anybody. If the signer does not want the signer to transfer the signature it can use undeniable signatures [5] or designated verifier signatures [17], but then the holder of the signature no longer holds any indisputable evidence of a signature. Chaum [4] proposed designated confirmer signatures (DC-signatures) as a means to get the best of both worlds at the price of the introduction of a semi-trusted third party called the confirmer.

An example application for DC-signatures is a job offer scenario. Alice is offered a job by Bob and wishes to receive a formal signed offer at some point, but Bob wants to avoid that Alice shows this offer to his competitor Eve. To solve the problem Carol comes to the rescue. Bob computes a DC-signature using his own secret key and Carols public key. Then he proves to Alice that he formed the signature in this way. The DC-signature is special in that it can only be verified directly by Carol, and its distribution is indistinguishable from a distribution that can be computed using only the public keys of Carol and Bob. Furthermore, given a valid/invalid DC-signature, Carol can either convert it into a valid/invalid ordinary signature of Bob that can be verified by anybody, or she

[⋆] This is an extended abstract. The full paper [23] is available at the Cryptology ePrint Archive, http://eprint.iacr.org.

can prove that she has the ability to do this. Bob can assume that nobody can forge a signature for his public key, and that as long as Carol is honest nobody learns that he signed an offer. Alice can safely assume that Bob can not fool her, and that if Bob denies having signed an offer and Carol is honest, then Carol can prove to anybody that Bob is lying.

1.1 Previous Work

The first formal model of DC-signatures was given by Okamoto [19], but he did not consider the problem of signer coercion. Thus, a signer could be coerced into confirming/denying a signature without the randomness of the signature computation. This problem was considered by Camenisch and Michels [2], who provided stronger definitions. They also proposed both a scheme based on general primitives and more practically oriented schemes, and sketched a security proof for the general construction. In their work on verifiable encryption of discrete logarithms Camenisch and Shoup [3] give a very brief sketch of a DC-signature scheme where most interactive protocols use Schnorr-style techniques.

Goldwasser and Waisbard [16] proposed a relaxed security definition to allow the proofs of knowledge to be strong witness hiding instead of zero-knowledge, and thus allow concurrency. They give a transformation that converts an ordinary signature scheme into a DC-signature scheme secure according to their relaxed definition. They use no random oracles, but the disavowal protocol is based on general zero-knowledge techniques, and the other protocols are based on cut-and-choose techniques.

Gentry, Molnar, and Ramzan [13] considered another relaxation based on an observation originally made by Michels and Stadler [18]. Instead of computing a signature of the message directly, the signer computes a "confirmer commitment" of the message, and then sign the commitment. The constructions in [13] are efficient and do not rely on the random oracle model, but they require the existence of trusted RSA-parameters.

1.2 Our Contributions

Firstly, we take a careful look at existing definitions of DC-signatures. It turns out that two protocols that are not mentioned in previous works, are needed for successful deployment: a proof of correct conversion of a signature, and a proof that a public key is "well formed". We also observe that the definitions of security proposed by Camenisch and Michels [2], Goldwasser and Waisbard [16], and Gentry et al. [13] respectively do not ensure unforgeability when the confirmer is malicious. Furthermore, we note that the relaxed definition in [16] does not prevent transferability, which is arguably a key property of DC-signatures, and the definition in [2] is flawed and can not be satisfied at all. Thus, previous definitions do not capture the notion of DC-signatures correctly. We propose new definitions that correct these deficiencies.

Secondly, we consider how to construct a secure DC-signature scheme. We prove the security of a generic construction with respect to the new security definition. We then describe an instantiation of the generic construction that is secure under the strong RSA-assumption, the decision composite residuosity assumption, and the decision Diffie-Hellman assumption. In contrast to the scheme briefly sketched by Camenisch and Shoup [3] our scheme does not rely on the random oracle model, and it satisfies stronger security requirements than the schemes proposed in [16] and [13]. Furthermore, the setting we consider is stricter in that we do not assume the existence of a trusted key generator as is done in [3]. Despite this our scheme is practical.

Thirdly, our approach to the problem of constructing DC-signatures is different from previous in that instead of relaxing the security definitions of DC-signatures, we relax the security definitions of the primitives used to construct them and prove that weaker primitives suffice. The relaxed notions we introduce and our techniques are of general interest in the construction of efficient and provably secure cryptographic schemes.

Because of space requirements we only sketch most of our results and proofs in this extended abstract, and focus on the new ideas presented. For details and proofs of all claims we refer the reader to the full version [23] of this paper.

1.3 Notation

We consider security with respect to *uniform* algorithms and our assumptions are also uniform in nature, but our results are easily translated to their non-uniform analogs. We use PT, PPT and EPPT to denote the set of uniform, uniform and probabilistic, and uniform expected, polynomial time Turing machines respectively. We let κ be the main security parameter. We write $\langle V(x), P(y)\rangle(z)$ to denote the output of V with private input x when it interacts with P with private input y on common input z. We write $A[S(x)]$ to denote that A has "oracle access" to an interactive Turing machine S with private input x. Formally, we assume that A has a separate pair of communication tapes over which it communicates with S. Whenever we write an expression of the form $\langle V(x), A[S(y)](z)\rangle(w)$ it is assumed that communication takes place in two phases. Before any message is communicated between V and A, A and S may communicate freely. Then some messages are communicated between V and A. When some message is again communicated between A and S, communication is no longer possible between V and A. Finally, when A chooses part of the common input on which it interacts with V, we write $\langle V, A[S(x)](y)\rangle(z, \cdot)$. We abuse notation and say that a protocol is an interactive proof if it is overwhelmingly sound and we say that a protocol is a proof of knowledge only if it is also an interactive proof.

We use 1 and 0, and logical true and false interchangeably. We denote the natural numbers by $\mathbb{N}$, the integers by $\mathbb{Z}$, the integers modulo n by $\mathbb{Z}_n$, the multiplicative group modulo n by $\mathbb{Z}_n^*$ and the subgroup of squares modulo n by SQ_n. We call a prime integer p safe if $(p-1)/2$ is prime.

We use the variation of the strong RSA-assumption which says that given a product N of two random safe primes of the same bit-size and a random $g \in SQ_N$, it is infeasible to compute (b, η) such that $b^\eta = g \bmod N$ and $\eta \neq \pm 1$. We use the variation of the decision composite residuosity assumption (DCR), which says that given a product n of two random safe primes of the same bitsize, it is infeasible to distinguish the uniform distribution on elements in $\mathbb{Z}_{n^2}^*$ from the uniform distribution on nth residues in $\mathbb{Z}_{n^2}^*$. We use the decision Diffie-Hellman assumption for the subgroup G_Q of squares of $\mathbb{Z}_P^*$, where $P = 2Q + 1$ is a safe prime. This says that if g generates G_Q and $\alpha, \beta, \gamma \in \mathbb{Z}_Q$ are random, then the distributions of $(g^\alpha, g^\beta, g^{\alpha\beta})$ and $(g^\alpha, g^\beta, g^\gamma)$ are indistinguishable.

2 Definition of Designated Confirmer Signature Schemes

A DC-signature scheme consists of algorithms and interactive protocols. There are two key generation algorithms $\mathsf{Kg}_\mathsf{s}^\mathsf{dc}$ and $\mathsf{Kg}_\mathsf{c}^\mathsf{dc}$ for the signer and confirmer respectively. There is a single signature algorithm Sig^dc that given a secret signature key, a message, and a confirmer public key outputs a signature. The signature can not be verified directly, but the confirmer can use a conversion algorithm Con^dc with his secret key and the signer public key to transform it into a signature that can be verified by anybody holding the public key of the signer using the verification algorithm Vf^dc. The protocols π_{wf} and π_c are used by the confirmer to prove that it formed its key correctly and the correctness of a conversion. The protocols π_v and π_e are used by the signer and confirmer respectively, to convince a verifier that a given DC-signature is valid and valid/invalid respectively. The protocols π_{wf} and π_c are not present in previous formalizations.

Definition 1 (DC Signature Scheme). *A designated confirmer signature scheme $\mathcal{DCS}$ consists of algorithms* $\mathsf{Kg}_\mathsf{s}^\mathsf{dc}, \mathsf{Kg}_\mathsf{c}^\mathsf{dc}, \mathsf{Sig}^\mathsf{dc} \in \mathrm{PPT}$ *and* $\mathsf{Con}^\mathsf{dc}, \mathsf{Vf}^\mathsf{dc} \in \mathrm{PT}$*, and interactive protocols* $\pi_{wf} = (P_{wf}, V_{wf})$*,* $\pi_c = (P_c, V_c)$*,* $\pi_v = (P_v, V_v)$*, and* $\pi_e = (P_e, V_e)$ *with the following completeness properties. For every* $\kappa \in \mathbb{N}$*, for every possible outputs* (ssk, spk) *of* $\mathsf{Kg}_\mathsf{s}^\mathsf{dc}(1^\kappa)$ *and* (sk, pk) *of* $\mathsf{Kg}_\mathsf{c}^\mathsf{dc}(1^\kappa)$ *respectively, for every* $m \in \{0,1\}^*$*, and for every* $r, \sigma_0 \in \{0,1\}^*$*, and with* $\sigma_1 = \mathsf{Sig}_{ssk,r}^\mathsf{dc}(m, pk)$

1. $\mathsf{Vf}_{spk}^\mathsf{dc}(m, \mathsf{Con}_{sk}^\mathsf{dc}(\sigma_1, spk)) = 1$,
2. $\Pr[\langle V_{wf}, P_{wf}(sk)\rangle(pk) = 1]$ *is overwhelming,*
3. $\Pr[\langle V_c, P_c(sk)\rangle(\sigma_0, \mathsf{Con}_{sk}^\mathsf{dc}(\sigma_0, spk), pk) = 1]$ *is overwhelming,*
4. $\Pr[\langle V_e, P_e(sk)\rangle(m, \sigma_0, \mathsf{Vf}_{spk}^\mathsf{dc}(m, \mathsf{Con}_{sk}^\mathsf{dc}(\sigma_0, spk)), pk, spk) = 1]$ *is overwhelming, and*
5. $\Pr[\langle V_v, P_v(ssk, r)\rangle(m, \sigma_1, pk, spk) = 1]$ *is overwhelming.*

2.1 Well-Formed Keys and Signatures

We introduce the notion of well-formed confirmer keys to formalize the set of strings which behave as keys functionally, and we introduce the notion of well-formed signatures as a generalization of the set of honestly generated signatures.

Definition 2 (Well-Formed Keys). *Let $\mathcal{DCS}$ be a DC-signature scheme. We say that the tuple* $(\mathsf{Kg}_{\mathsf{c},1}^\mathsf{dc}, \mathsf{Kg}_{\mathsf{c},2}^\mathsf{dc}, \mathsf{Kg}_{\mathsf{c},3}^\mathsf{dc})$ *splits* $\mathsf{Kg}_\mathsf{c}^\mathsf{dc}$ *if*

1. $\mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c},1}$ *and* $\mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c},2}$ *are probabilistic and* $\mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c},3}$ *is a deterministic polynomial time (in their first parameters) algorithms,*
2. *on input* 1^κ, $\mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c}}$ *computes* $pk_1 = \mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c},1}(1^\kappa)$, $sk = \mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c},2}(pk_1)$, *and* $pk_2 = \mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c},3}(1^\kappa, pk_1, sk)$, *and outputs* $(pk, sk) = ((pk_1, pk_2), sk)$,
3. *for every* $\kappa \in \mathbb{N}$ *and* $pk_1, pk_2 \in \{0,1\}^*$ *there exists at most one* $sk \in \{0,1\}^*$ *such that* $pk_2 = \mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c},3}(1^\kappa, pk_1, sk)$, *and*
4. *if* $pk_2 = \mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c},3}(1^\kappa, pk_1, sk)$, *then for every output* (spk, ssk) *of* $\mathsf{Kg}^{\mathsf{dc}}_{\mathsf{s}}(1^\kappa)$ *and* $m, r \in \{0,1\}^*$: $\mathsf{Vf}^{\mathsf{dc}}_{spk}(m, \mathsf{Con}^{\mathsf{dc}}_{sk}(\mathsf{Sig}^{\mathsf{dc}}_{ssk,r}(m, pk), spk)) = 1$.

We say that (pk_1, pk_2) *is well-formed with respect to the splitting if* $pk_2 = \mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c},3}(pk_1, sk)$ *for some* sk. *We say that* $((pk_1, pk_2), sk)$ *is well-formed for such a secret key* sk.

If the signer or the confirmer proves to the verifier that a DC-signature is valid/invalid relative a well-formed confirmer public key, then the verifier is confident that if converted, the result is also valid/invalid in a consistent way. Every key generator can be trivially split, but we are interested in splittings that given (pk_1, pk_2) allow a simple proof of knowledge of sk such that $((pk_1, pk_2), sk)$ is well-formed.

Definition 3 (Well-Formed Signature). *Let* $\mathcal{DCS}$ *be a DC-signature scheme and let* $wf : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ *be a polynomially computable function. We say that* wf *is a well-formedness function with respect to* $\mathcal{DCS}$ *and a splitting of* $\mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c}}$ *if for every well-formed* (pk, sk), *every output* (spk, ssk) *of* $\mathsf{Kg}^{\mathsf{dc}}_{\mathsf{s}}(1^\kappa)$, *and every possible output* $s = \mathsf{Con}^{\mathsf{dc}}_{sk}(\mathsf{Sig}^{\mathsf{dc}}_{ssk}(m, pk), spk)$, *we have* $wf(s, pk, spk) = 1$.

All honestly generated signatures are well formed, but some valid signatures may not be. It is trivial to see that there exists a well-formedness function for every DC-signature scheme, but we are interested in well-formedness functions that simplify the construction of our protocols.

2.2 Definition of Security

Assume that some splitting and well-formedness functions are fixed and define the following relations.

Definition 4 (Relations).

1. Well-Formed Confirmer Keys. *Denote by* $\mathcal{R}_{wf}$ *the set of all well-formed key pairs* $((pk_1, pk_2), sk)$.
2. Correct Conversion. *Denote by* $\mathcal{R}_c$ *the set of pairs* $((\sigma, s, pk, spk), sk)$ *such that* $(pk, sk) \in \mathcal{R}_{wf}$ *and* $s = \mathsf{Con}^{\mathsf{dc}}_{sk}(\sigma, spk)$.
3. Correct Evaluation. *Denote by* $\mathcal{R}_e$ *the set of pairs* $((m, \sigma, c, pk, spk), sk)$ *such that* $(pk, sk) \in \mathcal{R}_{wf}$, *and* $s = \mathsf{Con}^{\mathsf{dc}}_{sk}(\sigma, spk)$ *for some* s *such that* $\mathsf{Vf}^{\mathsf{dc}}_{spk}(m, s) = c$ *and* $wf(s, pk, spk) = 1$.

4. PROOF OF VALIDITY. *Denote by* $\mathcal{R}_v$ *the set of pairs* $((m, \sigma, pk, spk), w)$*, where* w *is a witness that* $\mathsf{Vf}^{\mathsf{dc}}_{spk}(m, \mathsf{Con}^{\mathsf{dc}}_{sk}(\sigma, spk)) = 1$ *for every* sk *such that* $(pk, sk) \in \mathcal{R}_{wf}$*.*

The relation $\mathcal{R}_e$ only considers *well-formed* signatures. If a signature is not well-formed, it was by definition computed by a corrupt signer. In this case the confirmer need not protect the signer and can simply convert the signature and prove that it did so correctly. The relation $\mathcal{R}_v$ does *not* capture the set of valid signatures. It captures the set of signatures that are valid *given that the public confirmer key is well-formed.*

We now consider what each honest party or group of honest parties in a DC-signature scheme might expect from a secure implementation.

Honest Verifier. An honest verifier naturally expects that it is infeasible to convince it of a false statement. Furthermore, it seems reasonable that the verifier only accepts a proof of well-formedness of a public key if it shows that the prover knows the secret key, since otherwise it can not be confident that the confirmer actually is able to convert a signature.

Definition 5 (Soundness). *A DC-signature scheme* $\mathcal{DCS}$ *is sound if* π_c*,* π_e*, and* π_v *are interactive proofs for the relations* $\mathcal{R}_c$*,* $\mathcal{R}_e$*, and* $\mathcal{R}_v$ *respectively, and* π_{wf} *is a proof of knowledge for the relation* $\mathcal{R}_{wf}$*.*

Note that if a confirmer public key pk is well-formed and σ is a candidate signature for a signer public key spk, then well-formedness implies that a signature can be converted in only one way. Thus, the confirmer can not choose if a signature should be considered valid or not. Well-formedness also implies that if a signer proves the validity of σ, it can not be converted into an invalid one.

It may be dangerous for a signer to use a confirmer public key that is not well-formed. Thus, we assume that any signer (or somebody the signer trusts) executes π_{wf} with the confirmer before using its public key.

Honest Signer. It must be infeasible for the adversary to convince anybody that the honest signer has signed a message m unless this is the case. This must hold even when the adversary can ask arbitrary signature queries and execute the proof of validity protocol π_v. In other words we need a slight generalization of security against chosen message attacks [15].

We formalize the honest signer S to be a probabilistic interactive Turing machine that accepts as input a key pair (spk, ssk). Whenever it receives a message with prefix π_{wf}, $\mathsf{Sig}^{\mathsf{dc}}$, or π_v on its communication tape it halts the execution of any executing interactive protocol and proceeds as follows. Given a message (π_{wf}, pk) it executes the verifier V_{wf} of the protocol π_{wf} on common input pk. If V_{wf} accepts the proof, then S stores pk. Given a message $(\mathsf{Sig}^{\mathsf{dc}}, m, pk)$ it checks if it has stored pk. If not, it returns $\perp$, and otherwise it computes $\sigma = \mathsf{Sig}^{\mathsf{dc}}_{ssk,r}(m, pk)$ and writes σ on the communication tape. Given a message (π_v, m, σ, pk), where it previously computed $\sigma = \mathsf{Sig}^{\mathsf{dc}}_{ssk,r}(m, pk)$ it executes the prover of the protocol π_v on common input (m, σ, pk, spk) and private input some witness w that $((m, \sigma, pk, spk), w) \in \mathcal{R}_v$. When we use several copies of S below we index them for easy reference.

Experiment 1 (CMA-Security, $\mathsf{Exp}^{\mathsf{cma}}_{\mathcal{DCS},A}(\kappa)$).

$$\begin{aligned}(spk, ssk) &\leftarrow \mathsf{Kg}^{\mathsf{dc}}_{\mathsf{s}}(1^\kappa)\\ (m, s) &\leftarrow A[S(spk, ssk)](spk)\end{aligned}$$

If $\mathsf{Vf}^{\mathsf{dc}}_{spk}(m, s) = 0$ or if S signed m return 0 and otherwise 1.

Definition 6 (CMA-Security). *A DC-signature scheme $\mathcal{DCS}$ is secure against chosen message attacks (CMA-secure) if for every $A \in \mathrm{PPT}$: $\Pr[\mathsf{Exp}^{\mathsf{cma}}_{\mathcal{DCS},A}(\kappa) = 1]$ is negligible.*

In the definitions of Camenisch and Michels [2], Goldwasser and Waisbard [16], and Gentry et al. [13] security hold only with respect to *honestly* generated confirmer keys, i.e., their definitions do not ensure any form of CMA-security for the signer when the confirmer is corrupted.

The definition does not say that an adversary can not form a bit-string σ and then convince an honest verifier using π_v or π_e that this is a valid designated signature of some message m not signed by S, but we prove in the full paper that this follows from soundness and CMA-security.

Honest Confirmer. Nobody except the confirmer should be able to play the role of the prover in the protocols π_{wf}, π_c, and π_e using the honest confirmers public key as common input, even after interacting with the real confirmer.

We formalize the honest confirmer C to be a probabilistic interactive Turing machine that accepts as input a key pair (pk, sk). Whenever it receives a message with prefix π_{wf}, $\mathsf{Con}^{\mathsf{dc}}$, or π_e on its input communication tape it halts the execution of any interactive protocol it is executing and proceeds as follows. Given a message (π_{wf}) it executes the prover of protocol π_{wf} on common input pk and private input sk. Given a message $(\mathsf{Con}^{\mathsf{dc}}, \sigma, spk)$ it computes $s = \mathsf{Con}^{\mathsf{dc}}_{sk}(\sigma, spk)$, writes s on its output communication tape, and executes the prover of protocol π_c on common input (σ, s, pk, spk) and private input sk. On input (π_e, m, σ, spk) it computes $s = \mathsf{Con}^{\mathsf{dc}}_{sk}(\sigma, spk)$. If $wf(s, pk, spk) = 1$, i.e., s is well-formed, then C computes $c = \mathsf{Vf}^{\mathsf{dc}}_{spk}(m, s)$, writes c on its output communication tape, and executes the prover of protocol π_e on common input (m, σ, c, pk, spk) and private input sk. Otherwise C writes $(\mathsf{malformed}, s)$ on its output communication tape and executes the prover of protocol π_c on common input (σ, s, pk, spk) and private input sk.

Experiment 2 (Impersonation-Resistance, $\mathsf{Exp}^{\mathsf{imp-res}}_{\mathcal{DCS},A}(\kappa)$).

$$\begin{aligned}(pk, sk) &\leftarrow \mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c}}(1^\kappa)\\ d_1 &\leftarrow \langle V_{wf}, A[C(pk, sk)](pk)\rangle(pk)\\ d_2 &\leftarrow \langle V_c, A[C(pk, sk)](pk)\rangle(\cdot, \cdot, pk, \cdot)\\ d_3 &\leftarrow \langle V_e, A[C(pk, sk)](pk)\rangle(\cdot, \cdot, \cdot, pk, \cdot)\end{aligned}$$

Return $d_1 \vee d_2 \vee d_3$.

Definition 7 (Impersonation Resistance). *A designated confirmer signature scheme $\mathcal{DCS}$ is impersonation-resistant if for every $A \in \mathrm{PPT}$:*
$\Pr[\mathsf{Exp}^{\mathsf{imp-res}}_{\mathcal{DCS},A}(\kappa) = 1]$ *is negligible.*

Note that C never invokes P_e without executing $\mathsf{Con}^{\mathsf{dc}}$. This is without loss of generality, since $\mathsf{Con}^{\mathsf{dc}}$ is deterministic and the common input contains the extracted signature anyway. Note that this differs from the signature case, where a signer could potentially want to prove the correctness of a particular signature to several receivers.

Remark 1. A stronger definition would allow the adversary to interact with the confirmer and the verifier concurrently on other inputs. Unfortunately, such a definition requires the protocols π_c and π_e to be non-malleable [11] with respect to each other and themselves. General methods such as [21] can be used to construct non-malleable zero-knowledge protocols, but currently these techniques are far from practical. Thus, we do not follow this definitional path.

Honest Signer and Honest Confirmer. To start with we observe that from the point of view of an honest signer and honest verifier, or from the point of view of an honest verifier and honest confirmer, no additional requirements are natural to impose.

When the signer and the confirmer are honest we require that knowledge that the signer signed a particular message can not be transfered. Note that this is needed in the job offer scenario. Non-transferability can clearly only hold until a DC-signature has been converted.

We formalize this as follows. Let SC be the machine that simulates both S and C on inputs (pk, sk) and (spk, ssk) except for the following modifications. Given a message $(\mathsf{Sig}^{\mathsf{dc}}, m, pk')$ with $pk' = pk$ it waits for a message (m, σ), stores this, and writes σ on its communication tape. If later invoked on (π_v, m, σ, pk) it returns $\perp$ instead of invoking P_v. For $pk' \neq pk$ it behaves as S. Given a message $(\mathsf{Con}^{\mathsf{dc}}, \sigma, spk')$ such that $spk' = spk$ and (m, σ) is stored it checks if (m, σ, s) is stored for some s. If not, then it computes $s = \mathsf{Con}^{\mathsf{dc}}_{sk}(\mathsf{Sig}^{\mathsf{dc}}_{ssk}(m, pk), spk)$ and stores (m, σ, s). Finally, it writes s on its communication tape. It does not invoke the prover of π_c. If $spk' \neq spk$ or (m, σ) is not stored it behaves as C. Finally, given a message (π_e, m, σ, spk), where (m', σ) is stored for some m' it returns 0 if $m \neq m'$ and 1 otherwise. It does not execute the prover of π_e.

Intuitively, SC *delays* the computation of every DC-signature using the public key pk until it is converted. We want to say that if there is an adversary A that interacts with S and C, there is another adversary A' that interacts with SC such that its output is indistinguishable from that of A, despite that all its signature queries are "delayed". For this to make sense the order of messages sent to S, C, and SC must be given to the distinguisher as well. We say that an adversary is scheduled if whenever it writes a message with prefix $\mathsf{Sig}^{\mathsf{dc}}$, π_v, π_{wf}, $\mathsf{Con}^{\mathsf{dc}}$, and π_e the message and any return value (excluding the messages exchanged by the protocols that may be invoked) are stored on a special write only scheduling tape. Furthermore, when the adversary halts its output is prefixed by its scheduling tape. The additional input pk to S below is stored as a well-formed public key, and this is done in the simulation of SC as well.

Experiment 3 (Non-Transferability, $\mathsf{Exp}^{\mathsf{non-trans}-b}_{\mathcal{DCS},A,V}(\kappa)$).

$$((pk, sk), (spk, ssk)) \leftarrow (\mathsf{Kg}^{\mathsf{dc}}_{\mathsf{c}}(1^\kappa), \mathsf{Kg}^{\mathsf{dc}}_{\mathsf{s}}(1^\kappa))$$
$$d \leftarrow \begin{cases} D(A(pk, spk, ssk)[SC(pk, sk, spk, ssk)]) & \text{if b=0} \\ D(A(pk, spk, ssk)[S(pk, spk, ssk), C(pk, sk)]) & \text{if b=1} \end{cases}$$

Definition 8 (Non-Transferability). *A DC-signature scheme $\mathcal{DCS}$ is non-transferable if for every scheduled $A_1 \in$ EPPT there exists a scheduled $A_0 \in$ EPPT such that for every distinguisher $D \in$ EPPT:*
$|\Pr[\mathsf{Exp}^{\mathsf{non-trans}-1}_{\mathcal{DCS},A_1,D}(\kappa) = 1] - \Pr[\mathsf{Exp}^{\mathsf{non-trans}-0}_{\mathcal{DCS},A_0,D}(\kappa) = 1]|$ *is negligible.*

Remark 2. Our definition is similar to the "liberal" definition of zero-knowledge in that the simulator is allowed to run in *expected* polynomial time.

Remark 3. Again, our experiment is not completely realistic. A stronger definition would allow the adversary to interact concurrently with S and C on other common inputs when trying to convince a verifier. Unfortunately, such definitions imply that the protocols π_{wf}, π_c, π_e, and π_v are non-malleable in a very strong sense. We are not aware of any general methods to achieve this.

Informally, a DC-signature scheme is coercion-free if a signer can reveal its secret signing key and still claim that it did not compute a particular DC-signature. Naturally, this can only hold as long as the DC-signature is not converted, or proved to be valid. Note that this is already captured in our definition of non-transferability.

The definition of non-transferability in Camenisch and Michels [2] can not be satisfied, since it requires the existence of a straight-line zero-knowledge simulator for an interactive proof without set-up assumptions. The definition of Goldwasser and Waisbard [16] only prevents the adversary from transferring confidence of validity of a signature using the confirmation protocol of the scheme. It says nothing about the possibility of using another confirmation protocol. The relaxed definition of Gentry et al. [13] explicitly allows some forms of transferability.

Most previous definitions require some form of indistinguishability of signatures computed by different signers, but this is unnecessarily strong. In any claim about a signature, the holder of the signature would disclose the identity of the claimed signer anyway, and our definition implies that anybody can generate something indistinguishable from a valid signature of any such signer.

Definition of Security. We now define security of a DC-signature scheme in the natural way.

Definition 9. *A designated confirmer signature scheme $\mathcal{DCS}$ is secure if it is sound, CMA-secure, impersonation-resistant, and non-transferable.*

On Concurrency. In our definitions the "oracle access" to the honest signer S and the honest confirmer C are sequential. Stronger definitions similar to those of Camenisch and Michels [2], where the adversary is given concurrent "oracle access", follow by giving the adversary access to several copies of S and C, each executing on the same input key pair.

3 Theoretical Tools

3.1 A Relaxation of Zero-Knowledge

The definition of zero-knowledge is very strong in that the simulation property must hold with respect to every verifier and *every instance* (x, w) in the relation $\mathcal{R}$ under consideration. As pointed out by Goldreich [14] it is quite natural to consider a uniform definition that only requires that it is *infeasible to find an instance* on which a verifier can gain knowledge.

In many cryptographic settings the instance can not be chosen completely freely by the adversary, e.g., the adversary may ask an honest party to prove that it performed a decryption correctly, where the keys to the cryptosystem are generated honestly. Furthermore, in some security proofs the simulator can be allowed an additional advice string *dependent* on the instance, e.g., if a decryption oracle is present in the environment where the simulator is invoked we may give the simulator decryptions of some ciphertexts. The following definition allows for both these settings.

Experiment 4 (Zero-Knowledge, $\mathsf{Exp}^{(T,F)-\mathsf{zk}-b}_{\pi,\mathcal{R},I,V^*,D}(\kappa)$).

$$\begin{aligned}
t &\leftarrow T(1^\kappa)\\
(i, z) &\leftarrow I(1^\kappa, t)\\
(x, w, a) &\leftarrow F(t, i)\\
d &\leftarrow \begin{cases} D(x, z, a, \langle V^*(z), P(w)\rangle(x)) & \text{if } \mathrm{b=0}\\ D(x, z, a, M(z, a, x)) & \text{if } \mathrm{b=1}\end{cases}
\end{aligned}$$

Return 0 if $\mathcal{R}(x, w) = 0$ and d otherwise.

Definition 10. *Let $\pi = (P, V)$ be an interactive protocol, let $T \in \mathrm{PPT}$ and $F : \{0,1\}^* \to \{0,1\}^*$, and let $\mathcal{R}$ be a relation. We say that π is (T, F)-zero-knowledge for $\mathcal{R}$ if for every verifier $V^* \in \mathrm{EPPT}$ there exists a simulator $M \in \mathrm{EPPT}$ such that for every instance chooser $I \in \mathrm{EPPT}$ and every distinguisher $D \in \mathrm{EPPT}$: $|\Pr[\mathsf{Exp}^{(T,F)-\mathsf{zk}-0}_{\pi,\mathcal{R},I,V^*,D}(\kappa) = 1] - \Pr[\mathsf{Exp}^{(T,F)-\mathsf{zk}-1}_{\pi,\mathcal{R},I,V^*,D}(\kappa) = 1]|$ is negligible.*

Remark 4. We do not require that F is polynomial time in the definition, but the concrete protocols we present are (T, F)-zero-knowledge with efficiently computable functions F. This seems also essential to allow sequential composition.

Remark 5. The definition makes sense for non-uniform adversaries as well. Furthermore, the definition can be both generalized and relaxed. One natural relaxation is to give only part of the sample t to the instance finder. Note that a probabilistic F is captured by this relaxation. A related definition gives the instance finder access to some specific oracle, i.e., we would talk about (T, F, O)-zero-knowledge for some specific oracle O. This seems to make most sense when the oracle is efficiently computable using the sample t (of which not all is given to the instance finder).

Choose some canonical interpretation of strings such that the output of I is always of the form $((x_1, w_1), z)$. Then when the output of T is always of the form $t = (x_2, w_2)$, and $F(t, i) = ((x_1, x_2), (w_1, w_2), \emptyset)$, we simply say that the protocol is T-zero-knowledge.

We show in the full paper that if a simulator satisfies the definition, then it can be used instead of a real protocol execution polynomially many times sequentially as long as F is polynomial time. This extension is necessary in our analysis.

3.2 Cryptosystems With Labels and Δ-CCA2-Security

Our starting point is the generalization of CCA2-security for cryptosystems with labels introduced by Shoup and Gennaro [22].

In such a scheme the encryption algorithm Enc takes as input a label L in addition to a public key pk and a message m. The decryption algorithm Dec takes as input a label L in addition to a secret key sk and a ciphertext c. CCA2-security is then defined as usual except that the adversary must choose a label L in addition to the two challenge ciphertexts m_0 and m_1, and it may not ask the decryption query (L, c), where c is the challenge ciphertext.

The definition of CCA2-security is strict in that the indistinguishability property of ciphertexts holds for *any* two messages. In our setting a weaker property suffices, namely that any two encrypted signatures from the *same* signer are indistinguishable. Thus, we introduce the following relaxed definition.

Experiment 5 (Δ-CCA2-Security, $\mathsf{Exp}_{\mathcal{CS},A}^{\Delta-\mathsf{cca2}-b}(\kappa)$).

$$\begin{aligned}(pk, sk) &\leftarrow \mathsf{CSKg}(1^\kappa)\\ (r, m_0, m_1, \mathrm{state}) &\leftarrow A^{\mathsf{Dec}_{sk}(\cdot,\cdot)}(\mathsf{choose}, pk)\\ c &\leftarrow \mathsf{Enc}_{pk}(\Delta(r, m_b))\\ d &\leftarrow A^{\mathsf{Dec}_{sk}(\cdot,\cdot)}(\mathsf{guess}, \mathrm{state}, c)\end{aligned}$$

Interpret $\Delta(r, m_b)$ as a pair (L, m_b'). The experiment returns 0 if $\mathsf{Dec}_{sk}(\cdot, \cdot)$ was queried on (L, c), and otherwise d.

Definition 11 (Δ-CCA2-Security). *Let $\Delta \in$ PPT. A public key cryptosystem $\mathcal{CS}$ with labels is said to be Δ-CCA2-secure if for every adversary $A \in$ PPT: $|\Pr[\mathsf{Exp}_{\mathcal{CS},A}^{\Delta-\mathsf{cca2}-0}(\kappa) = 1] - \Pr[\mathsf{Exp}_{\mathcal{CS},A}^{\Delta-\mathsf{cca2}-1}(\kappa) = 1]|$ is negligible in κ.*

3.3 Collision-Free Signature Schemes

We say that a signature scheme is collision-free if it is infeasible to find two distinct messages and a signature such that the signature is a valid with respect to both messages, even if the adversary is given the honestly generated secret key and the public key.

4 A Generic Construction of Designated Confirmer Signatures

It is natural to construct DC-signatures from a CMA-secure signature scheme and a CCA2-secure cryptosystem. A signer holds a secret key for the signature scheme and the confirmer holds a secret key of the cryptosystem. A DC-signature is simply an ordinary signature encrypted with the cryptosystem, conversion corresponds to decryption, and zero-knowledge proofs of knowledge are used to instantiate the protocols. The theorem below implies that this is secure, but for most signature schemes and cryptosystems it is prohibitively inefficient.

As a first step in the construction of an *efficient* DC-signature scheme we prove that weaker primitives suffice to construct a secure DC-signature scheme, but the basic idea is the same. Let $\mathcal{CS} = (\mathsf{CSKg}, \mathsf{Enc}, \mathsf{Dec})$ be a cryptosystem with labels and let $\mathcal{SS} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf})$ be a signature scheme with fixed size signatures (this is easy to ensure by padding) that fit in the plaintext space of $\mathcal{CS}$. Define $\mathsf{Kg}_{\mathsf{s}}^{\mathsf{dc}}$ to compute $(spk, ssk) = \mathsf{Kg}(1^\kappa)$ and $s_\perp = \mathsf{Sig}_{ssk}(\perp)$, where $\perp$ is a special symbol, and output $((spk, s_\perp), ssk)$ (we drop $s_\perp$ from our notation when convenient). Define $\mathsf{Kg}_{\mathsf{c}}^{\mathsf{dc}}(1^\kappa)$ to output $\mathsf{CSKg}(1^\kappa)$, and define $\mathsf{Vf}^{\mathsf{dc}}$ to be Vf except that $\mathsf{Vf}_{spk}^{\mathsf{dc}}(\perp, \cdot) = 0$. On input ssk, m, and pk the DC-signature algorithm $\mathsf{Sig}^{\mathsf{dc}}$ is defined to compute $s = \mathsf{Sig}_{ssk}(m)$ and $\sigma = \mathsf{Enc}_{pk}(spk, s)$, and output σ. On input sk, σ, and spk the conversion algorithm $\mathsf{Con}^{\mathsf{dc}}$ is defined to output $s = \mathsf{Dec}_{sk}(spk, \sigma)$. In other words, the public signature key spk is used as a label. Let $(\mathsf{Kg}_{\mathsf{c},1}^{\mathsf{dc}}, \mathsf{Kg}_{\mathsf{c},2}^{\mathsf{dc}}, \mathsf{Kg}_{\mathsf{c},3}^{\mathsf{dc}})$ be a splitting of $\mathsf{Kg}_{\mathsf{c}}^{\mathsf{dc}}$ and let wf be some well-formedness function with respect to $\mathcal{DCS}$. Let π_{wf}, π_c, π_e, and π_v be interactive protocols, complete with respect to the relations $\mathcal{R}_{wf}$, $\mathcal{R}_c$, $\mathcal{R}_e$, and $\mathcal{R}_v$. It is easy to see that $\mathcal{DCS} = (\mathsf{Kg}_{\mathsf{s}}^{\mathsf{dc}}, \mathsf{Kg}_{\mathsf{c}}^{\mathsf{dc}}, \mathsf{Con}^{\mathsf{dc}}, \mathsf{Vf}^{\mathsf{dc}}, \pi_{wf}, \pi_c, \pi_e, \pi_v)$ is a DC-signature scheme.

The algorithm $\Delta(r, (r', m))$ first computes $(spk, ssk) = \mathsf{Kg}_r(1^\kappa)$ and $s = \mathsf{Sig}_{ssk,r'}(m)$, and then outputs (spk, s). Define $T_{hs}(1^\kappa) = (\mathsf{Kg}_{\mathsf{c}}^{\mathsf{dc}}(1^\kappa), \mathsf{Kg}_{\mathsf{s}}^{\mathsf{dc}}(1^\kappa))$. Define F_{hs} to take as input the pair $((pk, sk, spk, ssk), (r, m, m'))$, compute $\sigma = \mathsf{Sig}_{ssk,r}^{\mathsf{dc}}(m)$, and output $((m', \sigma, \mathsf{Vf}_{spk}^{\mathsf{dc}}(m', \mathsf{Con}_{sk}^{\mathsf{dc}}(\sigma)), pk, spk), sk, \emptyset)$. Let $T_{cs}(1^\kappa)$ simply output $\mathsf{Kg}_{\mathsf{c}}^{\mathsf{dc}}(1^\kappa)$. Define F_{cs} to take input $((pk, sk), (m, \sigma, c, spk))$ and output the tuple $((m, \sigma, c, pk, spk), sk, \mathsf{Con}_{sk}^{\mathsf{dc}}(\sigma, spk))$.

Theorem 1. [1] *Suppose that $\mathcal{CS}$ is Δ-CCA2-secure and that $\mathcal{SS}$ is CMA-secure and collision-free, and that π_{wf}, π_c, π_e, and π_v are proofs of knowledge for the relations $\mathcal{R}_{wf}$, $\mathcal{R}_c$, $\mathcal{R}_e$, and $\mathcal{R}_v$. Suppose that π_{wf} and π_c are $\mathsf{Kg}_{\mathsf{c}}^{\mathsf{dc}}$-zero-knowledge*

[1] Camenisch and Michels [2] claim a similar, but weaker, theorem according to their definition, but as explained above their definition can not be satisfied, and only a proof sketch is given.

for the relations $\mathcal{R}_{wf}$ and $\mathcal{R}_c$ respectively. Suppose that π_e is both (T_{hs}, F_{hs})-zero-knowledge and (T_{cs}, F_{cs})-zero-knowledge for the relation $\mathcal{R}_e$. Suppose π_v is $\mathsf{Kg}_{\mathsf{s}}^{\mathsf{dc}}$-zero-knowledge for the relation $\mathcal{R}_v$. Then $\mathcal{DCS}$ is secure.

4.1 On the Use of Two Distinct Weak Simulators

Perhaps the most interesting of our techniques is the use of two simulators for the same protocol, of which one requires additional advice. Consider the problem of constructing a black box-reduction of a successful attacker A against non-transferability into a successful attacker A' against the Δ-CCA2-security of the underlying cryptosystem. The Δ-CCA2-attacker A' takes a public key pk as input and must simulate the non-transferability experiment to the adversary without using the secret key sk. At some point A' outputs a random bit string r and two messages (r_0, m_0) and (r_1, m_1) to the Δ-CCA2-experiment, and it is given a ciphertext $\sigma = \mathsf{Enc}_{pk}(spk, \mathsf{Sig}_{ssk,r_b}(m_b))$ for a random $b \in \{0,1\}$, where $(spk, ssk) = \mathsf{Kg}_r(1^\kappa)$. The ciphertext σ is used in the simulation somehow, and finally A' outputs a bit. The simulation involves converting signatures, but A' may use its decryption oracle to answer such queries, as long as it never asks for a decryption of σ.

We observe that when the protocol π_e is simulated for some DC-signature $\sigma' \neq \sigma$ computed by A, the simulator is free to invoke the decryption oracle on σ', i.e., a (T_{cs}, F_{cs})-zero-knowledge simulator is sufficient. On the other hand, for the particular signature σ we can not proceed in this way, since that would violate the rules of the Δ-CCA2-experiment, but since σ is computed honestly using honestly formed signature keys a (T_{hs}, F_{hs})-zero-knowledge simulator suffices.

5 Concrete Tools

In this section we present the tools we need to instantiate the generic DC-signature scheme with an efficient concrete scheme under standard complexity assumptions.

5.1 A Twin-Moduli Signature Scheme

To prove the existence of the scheme presented below we must assume that an arbitrary bit-string can be embedded into a prime in an efficient way. We assume that there is an efficient algorithm $\mathsf{Emb}_f^{f'}$ that given $n \in [0, 2^\kappa - 1]$ with overwhelming probability finds $s \in [2^{f(\kappa)-1}, 2^{f(\kappa)-1} + 2^{f(\kappa)-f'(\kappa)} - 1]$ such that $e = 2^{f(\kappa)}n + s$ is prime. We call this assumption the (f, f')-Embedding Assumption. In practice this is not a problem for reasonable f and f'. The twin-moduli signature scheme, $\mathcal{SS}^2 = (\mathsf{Kg}^2, \mathsf{Sig}^2, \mathsf{Vf}^2)$, is based on using two sets of RSA-parameters and the embedding algorithm $\mathsf{Emb}_f^{f'}$. Denote by κ_r a security parameter such that $2^{-\kappa_r}$ is negligible in κ. On input 1^κ the key generator Kg^2 chooses $\kappa/2$-bit safe primes p_0, q_0, p_1, and q_1 randomly, defines

$N_0 = p_0 q_0$ and $N_1 = p_1 q_1$, chooses $g_0 \in SQ_{N_0}$ and $g_1 \in SQ_{N_1}$ randomly, and outputs $((N_0, g_0, N_1, g_1), (p_0, q_0, g_0, p_1, q_1, g_1))$. Set $\kappa_p = f(2\kappa_r + \kappa_m + 1)$ and $\kappa'_p = f'(2\kappa_r + \kappa_m + 1)$. The signature algorithm Sig^2 takes as input a private key $(p_0, q_0, g_0, p_1, q_1, g_1)$ and a message $m \in [0, 2^{\kappa_m} - 1]$. It chooses $r \in [2^{2\kappa_r + \kappa_m}, 2^{2\kappa_r + \kappa_m} + 2^{\kappa_r + \kappa_m} - 1]$ randomly. Then it computes

$$s_0 = \mathsf{Emb}_f^{f'}(r + m) \qquad e_0 = 2^{\kappa_p}(r + m) + s_0 \qquad z_0 = g_0^{1/e_0} \bmod N_0$$
$$s_1 = \mathsf{Emb}_f^{f'}(r) \qquad e_1 = 2^{\kappa_p} r + s_1 \qquad z_1 = g_1^{1/e_1} \bmod N_1 \ .$$

Finally, it outputs (r, z_0, s_0, z_1, s_1). The verification algorithm Vf^2 takes as input a public key (N_0, g_0, N_1, g_1), a message $m \in \{0,1\}^{\kappa_m}$, and a candidate signature (r, z_0, s_0, z_1, s_1). It computes $e_0 = 2^{\kappa_p}(r+m)+s_0$ and $e_1 = 2^{\kappa_p} r + s_1$, and verifies that $r \in [1, 2^{2\kappa_r + \kappa_m + 1} - 1]$, $s_0, s_1 \in [0, 2^{\kappa_p} - 1]$, that e_0 and e_1 are odd, and that $z_0^{e_0} = g_0 \bmod N_0$ and $z_1^{e_1} = g_1 \bmod N_1$. The basic idea of the scheme is similar to an idea of Cramer et al. [7], but the proposition below does not follow from their work. Using a collision-free hash function $H : \{0,1\}^* \to [0, 2^{\kappa_m} - 1]$ it can be used to sign messages of any length.

Proposition 1. *The scheme exists under the (f, f')-Embedding assumption and is CMA-secure and collision-free under the strong RSA-assumption.*

5.2 The Cramer-Shoup-Pailler Cryptosystem

The cryptosystem we use is based on Cramer and Shoup's [8] CCA2-secure version of the Paillier [20] as described in [3], i.e., it is a cryptosystem with labels. This cryptosystem is special in that plaintexts "live in the exponent", which simplifies the construction of Schnorr-like proofs about the plaintext.

If we think of our scheme as the combination of a Paillier ciphertext and a hash proof and write Enc for Paillier encryption we may explain the cryptosystem we use as follows. An encryption of a twin-moduli signature (r, z_0, z_1, s_0, s_1) essentially consists of a tuple

$$(E_a, E_{r_0}, c'_0, E_{r_1}, c'_1) = (\mathsf{Enc}(\mathsf{Pack}(r, s_0, s_1)), \mathsf{Enc}(r_0), g_0^{r_0} z_0, \mathsf{Enc}(r_1), g_1^{r_1} z_1) \ ,$$

where r_0 and r_1 are random and Pack is an invertible function that can be computed using only multiplication by constants and addition. Decryption is done in the obvious way. Note that $g_0^{r_0} z_0$ and $g_1^{r_1} z_1$ leaks information about the public key of the twin-moduli signature scheme, but encryptions of any two signatures of the same signer are indistinguishable. A single hash proof ties the components of a ciphertext together and the result is Δ-CCA2-secure when Δ is defined using the twin-moduli signature scheme.

5.3 Proofs of Knowledge of Equality Relations

There are various protocols in the literature [12, 1, 9, 3] for proving equality of integer exponents over groups of unknown order based on variations of Fujisaki-Okamoto commitments, under the strong RSA-assumption. These protocols are strictly speaking not proofs of knowledge, since extraction may fail with negligible probability over the choice of commitment parameters, but they can be used as proofs of knowledge provided that there are trusted commitment parameters present during the execution of the protocols. Furthermore, they are usually given as honest verifier zero-knowledge protocols, but it is easy to make them zero-knowledge for a malicious verifier. A useful feature of these protocols is that they bound the bit-size of the exponents.

5.4 Verifiable Generation of Hiding Commitment Scheme

The problem with the proofs of equal integer exponents in a two party setting is that it is difficult to generate the Fujisaki-Okamoto commitment parameters efficiently. Recall that the commitment parameters consist of a random RSA-modulus $N = pq$, where p and q are safe primes, a random $g \in SQ_N$, and $h = g^x$ for a random $x \in [0, N2^{\kappa_r}]$. A commitment C of $m \in \mathbb{Z}$ is formed as $C = g^r h^m$ for an $r \in [0, N2^{\kappa_r}]$. The problem is that if (N, g, h) are generated by the prover, then the commitments are not binding. On the other hand, if they are generated by the verifier, then h may not be of the form g^x, and then the commitments are not hiding. As far as we know there is no truly efficient solution to this problem.

We now sketch our solution to this problem. The prover generates a pair (N_r, g_r), where N_r is an RSA-modulus of two safe primes and $g_r \in SQ_{N_r}$. This is done only once and is part of the public key of the prover in our application. The verifier then generates (N, g, h) as above, except that it defines $h = g^x$ for a random $x \in [0, NN_r2^{\kappa_r}]$. It then computes a "commitment" $h_r = g_r^x$ of x and executes a Schnorr-like zero-knowledge "proof of knowledge" that the same integer x was used for both h_r and h. Extraction of x is possible with high probability provided that (N_r, g_r) are chosen correctly. This ensures that the parameters (N, g, h) can be used safely by a prover. On the other hand, provided $|SQ_{N_r}|$ and $|SQ_N|$ are relatively prime, h_r is essentially independently generated from h. This implies that the parameters (N, g, h) can be used safely by the verifier, since essentially no knowledge of x is leaked. To ensure that $|SQ_{N_r}|$ and $|SQ_N|$ are relatively prime with overwhelming probability we assume that N is generated independently from N_r. In practice this is very reasonable.

In the full paper we show that the parameters (N, g, h) output by the protocol below can be used to execute the proof of equal exponents that assumes trusted commitment parameters. We call the two players in the proof the generator G and the receiver R to distinguish them from their roles in a larger protocol. Denote by $\pi_{pl} = (P_{pl}, V_{pl})$ the zero-knowledge proof of knowledge of a logarithm for prime order groups described by Cramer et al. [6].

Protocol 1 (Secure Generation of Integer Commitment Scheme).
Common Input: A κ-bit integer N_r and $g_r \in \mathbb{Z}_{N_r}^*$ to both parties.

1. The receiver chooses a safe κ-bit prime $P = 2Q + 1$, and $H \in G_Q$ randomly, where G_Q is the unique subgroup of order Q. Then it chooses $z \in \mathbb{Z}_Q$ randomly, computes $K = H^z \bmod P$, hands (P, H, K) to the generator, and

executes π_{pl} as the prover on common input (P, H, K) and private input z. If the verifier rejects, then the generator hands $\perp$ to the receiver and halts.
2. The generator verifies that P is a safe prime and that $H, K \in G_Q$. Then it chooses an RSA-modulus N, $g \in SQ_N$, and $x \in [0, 2^{2\kappa+\kappa_r} - 1]$ randomly, and computes $h = g^x \bmod N$ and $h_r = g_r^x \bmod N_r$. Then it chooses $c_g \in [0, 2^{\kappa_c} - 1]$, $r_g \in \mathbb{Z}_q$, and $r \in [0, 2^{2\kappa+2\kappa_r+\kappa_c} - 1]$ randomly, defines $w = H^{r_g}K^{c_g}$, $\alpha = g^r \bmod N$ and $\alpha_r = g_r^r \bmod N_r$, and hands $(h, h_r, w, \alpha, \alpha_r)$ to the receiver.
3. The receiver chooses $c_r \in [0, 2^{\kappa_c} - 1]$ randomly and hands c_r to the generator.
4. The generator computes $c = c_g \oplus c_r$ and $d = cx + r \bmod 2^{2\kappa+2\kappa_r+\kappa_c}$, hands (d, c_g, r_g) to the receiver, and outputs (N, g, h).
5. The receiver outputs (N, g, h) if $H^{r_g}K^{c_g} = w$, $h^c\alpha = g^d \bmod N$ and $h_r^c\alpha_r = g_r^d \bmod N_r$. Otherwise it outputs $\perp$.

We denote the above protocol by $\pi_{tp} = (G, R)$, where G is the generator and R the receiver.

Remark 6. Although the receiver can use the same prime P in every protocol instance, the generator must check that P is of the expected form to be confident that it can run the protocol π_{pl}, which is only sound if G_Q has prime order.

Checking for primality is expensive, i.e., it requires $O(\kappa_r)$ exponentiations. If one assumes that it is infeasible to find a specific safe prime P such that the discrete logarithm problem is feasible in G_Q, then any party can choose a prime P that is used in every protocol instance. Then each party performs the primality test only once. This is a natural assumption in practice, where one can use a prime from a cryptographic standard.

Proposition 2. *For every pair (N_r, g_r) with $N_r \in \mathbb{N}$ and $g_r \in \mathbb{Z}_{N_r}^*$ the probability $\Pr[\langle R, G\rangle(N_r, g_r) \neq \perp]$ is overwhelming.*

Proposition 3. *Let (N, g, h) be randomly distributed Fujisaki-Okamoto parameters. Define $[R^*, G](N_r, g_r)$ to be a pair consisting of the output of R^* and G.*

Then for every receiver $R^* \in$ EPPT there exists a simulator $M \in$ EPPT such that for every pair (N_r, g_r) with $N_r \in \mathbb{N}$ and $g_r \in \mathbb{Z}_{N_r}^*$ the distributions of $[R^*, G](N_r, g_r)$ and $M(N_r, g_r, N, g, h)$ are statistically indistinguishable and $M(N_r, g_r, N, g, h)$ is always on the form $(\cdot, out_G)$ with $out_G \in \{(N, g, h), \perp\}$.

Informally, this simply means that we can simulate the protocol in such a way that a particular set of parameters are used. Since the generator does not have any secret input, it is not meaningful to say that the protocol is zero-knowledge. However, one may view the proposition as saying that the protocol leaks no knowledge to the receiver about the exponent x that is chosen by the generator within the protocol. In this sense the protocol is zero-knowledge.

Denote by T_{srsa} the algorithm that on input 1^κ outputs (N_r, g_r), such that N_r is a product of two random safe $\kappa/2$-bit safe primes and g_r is randomly chosen in SQ_{N_r}.

Proposition 4. *Suppose that $(N_r, g_r) = T_{srsa}(1^\kappa)$. Then the probability that the receiver outputs (N, g, h), where h is not in the subgroup of $\mathbb{Z}_N^*$ generated by g, is negligible under the strong RSA-assumption and the discrete logarithm assumption.*

Remark 7. Even with the modification of Remark 6 the protocol requires a non-constant number of exponentiations, since the generator may have to generate a new RSA-modulus to ensure that its modulus is generated independently of N_r. If the reader find this annoying, please note that if the generator chooses an RSA-modulus N with at least $2\kappa + 2$ bits it can reuse the *same* modulus in any proof, since the orders of g and g_r are coprime. However, the size of the random exponents in the protocol above, and in all protocols that use the modulus must then be doubled, and this gives a far less efficient protocol in practice. Thus, we detail the solution above.

6 An Efficient Instantiation

In the full paper we show that there is an instantiation of the generic DC-signature scheme which is secure under the DCR-assumption, the strong RSA-assumption, and the DDH-assumption. We sketch this solution below.

Given the generic description in Section 4 and Theorem 1 all the algorithms of our instantiation follow from setting the signature scheme equal to our twin-moduli signature scheme, and the Δ-CCA2-secure scheme equal to our variation of the Cramer-Shoup-Paillier scheme, provided that we define a splitting of the key generator and a well-formedness function. The public key of the cryptosystem contains an RSA-modulus n, but the scheme functions properly for any *integer* n with some minor modifications. The key generator outputs a commitment based on the El Gamal cryptosystem of the secret key that is unconditionally committing to ensure that the uniqueness property of well-formedness. Thus, we define the splitting such that it is not necessary to execute an expensive proof that n is correctly formed. The well-formedness function for signatures is based on the fact that for any honestly computed signature (r, z_0, z_1, s_0, s_1) it holds that $r \in [2^{2\kappa_r+\kappa_m}, 2^{2\kappa_r+\kappa_m} + 2^{\kappa_r+\kappa_m} - 1]$, $s_0, s_1 \in [2^{\kappa_p-1}, 2^{\kappa_p-1} + 2^{\kappa_p-\kappa_p'} - 1]$, and $z_1^{e_1} = g_1 \bmod N_1$. Recall that the signature verification algorithm only requires that $r \in [1, 2^{2\kappa_r+\kappa_m+1} - 1]$ and $s_0, s_1 \in [0, 2^{\kappa_p} - 1]$. The slack is exploited to avoid costly proofs of membership in intervals.

The idea of the twin-moduli signature scheme is loosely speaking that all that is needed to verify a signature can be done "in the exponent". Recall that a verification involves multiplication by constants, adding, checking for interval-membership and oddity, and then checking the roots of the signature. Let us write $C(m)$ for a Fujisaki-Okamoto commitment of the form $g^l h^m \bmod N$ for some random l, and simply write $\mathsf{Enc}(m)$ for a Paillier-part of a cryptotext, i.e., we ignore the encoding and the third component that guarantees CCA2-security. Then the proof of validity of a signature can be explained as follows. A DC-signature essentially consists of a tuple $(E_a, E_{r_0}, c_0', E_{r_1}, c_1')$ on the form

$$(\mathsf{Enc}(\mathsf{Pack}(r, s_0, s_1)), \mathsf{Enc}(r_0), g_0^{r_0} z_0, \mathsf{Enc}(r_1), g_1^{r_1} z_1) \ .$$

The prover forms commitments

$$\begin{aligned}C'_r &= C(r - 2^{2\kappa_r+\kappa_m})\\ C'_{s_0} &= C((s_0 - 2^{\kappa_p-1} - 1)/2)\\ C'_{s_1} &= C((s_1 - 2^{\kappa_p-1} - 1)/2)\end{aligned}$$

and proves knowledge of the committed values. The protocol used to do this also implies that $r - 2^{2\kappa_r+\kappa_m} \in [-2^{2\kappa_r+\kappa_m} + 1, 2^{2\kappa_r+\kappa_m} - 1]$ and $(s_0 - 2^{\kappa_p-1} - 1)/2, (s_1 - 2^{\kappa_p-1} - 1)/2 \in [-2^{\kappa_p-2} + 1, 2^{\kappa_p-2} - 1]$. Then the verifier computes

$$\begin{aligned}C_r &= C'_r C(2^{2\kappa_r+\kappa_m})\\ C_{s_0} &= (C'_{s_0})^2 C(2^{\kappa_p-1} + 1)\\ C_{s_1} &= (C'_{s_1})^2 C(2^{\kappa_p-1} + 1)\\ C_a &= C_r^{2^{2\kappa_p}} C_{s_0}^{2^{\kappa_p}} C_{s_1} \quad ,\end{aligned}$$

and the prover proves that the value committed to in C_a equal the value encrypted in E_a. Note that C_{s_0} and C_{s_1} are commitments to odd integers s_0 and s_1 in $[0, 2^{\kappa_p} - 1]$ and C_r is a commitment to an integer $r \in [1, 2^{2\kappa_r+\kappa_m+1} - 1]$. Thus, part of the verification has already been executed.

To complete the verification the verifier computes commitments of the integers e_0 and e_1 induced by the values r, s_0, and s_1 by forming

$$C_{e_0} = (C_r C(m))^{2^{\kappa_p}} C_{s_0} \quad \text{and} \quad C_{e_1} = C_r^{2^{\kappa_p}} C_{s_1} \quad .$$

All that then remains is to prove that if e_0 and e_1 are committed to in C_{e_0} and C_{e_1} and r_0 and r_1 are encrypted in E_{r_0} and E_{r_1}, then

$$(c'_0)^{e_0} = g_0^{e_0 r_0} g_0 \bmod N_0 \quad \text{and} \quad (c'_1)^{e_1} = g_1^{e_1 r_1} g_1 \bmod N_1 \quad .$$

This shows that the encrypted signature is a valid twin-moduli signature.

The proof of invalidity for well-formed signatures is similar, but more complicated in that at some point the prover must show that $z_0^{e_0}/g_0 \neq 1$ without revealing this value. A standard trick to solve this problem is to randomize the result, i.e., revealing $(z_0^{e_0}/g_0)^l$ for a randomly chosen l. However, in general it may happen that $z_0^{e_0}/g_0$ is contained in some particular subgroup of $\mathbb{Z}^*_{N_0}$ and the simulator clearly does not know if this is the case.

When the public signature key is chosen honestly and the malicious verifier does not know the factorization of N_0, it is infeasible to find any element that generates a non-trivial subgroup of SQ_{N_0}. Thus, in this case the above idea works straightforwardly and there is no problem. In other words we have a (T_{hs}, F_{hs})-zero-knowledge simulator.

For maliciously generated N_0, g_0, z_0, and e_0 the above approach does not work at all, and it seems difficult to come up with an efficient approach that does work. Fortunately, we know that it suffices to have a simulator that is given the values z_0 and e_0 as an additional advice string, and given these it is obviously trivial to generate $(z_0^{e_0}/g_0)^l$ with the right distribution. In other words we have a (T_{cs}, F_{cs})-simulator.

The complexity of our scheme for some practical parameters is given below.

Operation	Alg./Prot.	Signer	Confirmer	Verifier
Signing	$\mathsf{Sig}^{\mathsf{dc}}$	140		
Converting	$\mathsf{Con}^{\mathsf{dc}}$		66	
Verifying	$\mathsf{Vf}^{\mathsf{dc}}$			1
Well-Formedness	π_{wf}		61	59
Correctness of conversion	π_c		327	227
Validity/Invalidity	π_e		189	169
Validity	π_v	166		151

Table 1. The estimated average complexity of the algorithms and the protocols in terms of κ-bit exponentiations when $\kappa = 1024$, $\kappa_r = \kappa_c = 50$, and $\kappa_m = 160$.

7 Acknowledgments

I thank Ronald Cramer and Ivan Damgård for answering my questions about their work, I thank Dominik Raub and Stefano Tessaro for discussions on non-transferability, and I thank the anonymous reviewers of TCC 2007 for helpful comments.

References

1. F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.
2. J. Camenisch and Markus Michels. Confirmer signature schemes secure against adaptive adversaries. In *Advances in Cryptology – Eurocrypt 2000*, Lecture Notes in Computer Science, pages 243–258. Springer Verlag, 2000.
3. J. Camensisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology – Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer Verlag, 2003.
4. D. Chaum. Designated confirmer signatures. In *Advances in Cryptology – Eurocrypt '94*, volume 950 of *Lecture Notes in Computer Science*, pages 86–91. Springer Verlag, 1994.
5. D. Chaum and H. van Antwerpen. Undeniable signatures. In *Advances in Cryptology – Crypto '89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer Verlag, 1990.
6. R. Cramer, I. Damgård, and P. D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *Public Key Cryptography – PKC 2000*, volume 1751, pages 354–372. Springer Verlag, 2000.
7. R Cramer, I. Damgård, and T. P. Pedersen. Efficient and provable security amplifications. In *Security Protocols, International Workshop, Cambridge, United Kingdom, April 10-12, 1996, Proceedings*, volume 1189 of *Lecture Notes in Computer Science*, pages 101–109. Springer, 1996.
8. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. http://homepages.cwi.nl/ cramer/, June 1999.

9. I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer Verlag, 2002.
10. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
11. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *23rd ACM Symposium on the Theory of Computing (STOC)*, pages 542–552. ACM Press, 1991.
12. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer Verlag, 1997.
13. C. Gentry, D. Molnar, and Z. Ramzan. Efficient designated confirmer signatures without random oracles or zero-knowledge proofs (extended abstract). In *Advances in Cryptology – Asiacrypt 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 662–681. Springer Verlag, 2005.
14. O. Goldreich. A uniform-complexity treatment of encryption and zeroknowledge. *Journal of Cryptology*, 6(1):21–53, 1993.
15. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
16. S. Goldwasser and E. Waisbard. Transformation of digital signature schemes into designated confirmer signatures. In *1st Theory of Cryptography Conference (TCC)*, volume 2951 of *Lecture Notes in Computer Science*, pages 77–100. Springer Verlag, 2004.
17. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology – Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer Verlag, 1996.
18. M. Michels and M. Stadler. Generic constructions for secure and efficient confirmer signature schemes. In *Advances in Cryptology – Eurocrypt 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 406–421. Springer Verlag, 1998.
19. T. Okamoto. Designated confirmer signatures and public key encryption are equivalent. In *Advances in Cryptology – Crypto '94*, volume 839 of *Lecture Notes in Computer Science*, pages 61–74. Springer Verlag, 1994.
20. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Verlag, 1999.
21. R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In *37th ACM Symposium on the Theory of Computing (STOC)*, pages 533–542. ACM Press, 2005.
22. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology – Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 1–16. Springer Verlag, 1998.
23. Douglas Wikström. Designated confirmer signatures revisited. Cryptology ePrint Archive, Report 2006/123, 2006. http://eprint.iacr.org/.