On the Generic and Efficient Constructions of Secure Designated Confirmer Signatures

Guilin Wang[1], Joonsang Baek[1], Duncan S. Wong[2], and Feng Bao[1]

[1] Institute for Infocomm Research (I^2R)
21 Heng Mui Keng Terrace, Singapore 119613
{glwang, jsbaek, baofeng}@i2r.a-star.edu.sg
[2] City University of Hong Kong, Hong Kong
duncan@cityu.edu.hk

Abstract. For controlling the public verifiability of ordinary digital signatures, designated confirmer signature (DCS) schemes were introduced by Chaum at Eurocrypt 1994. In such schemes, a signature can be verified only with the help of a semi-trusted third party, called the designated confirmer. The confirmer can further selectively convert individual designated confirmer signatures into ordinary signatures so that anybody can check their validity. In the last decade, a number of DCS schemes have been proposed. However, most of those schemes are either inefficient or insecure. At Asiacrypt 2005, Gentry, Molnar and Ramzan presented a generic transformation to convert any signature scheme into a DCS scheme, and proved the scheme is secure in their security model. Their DCS scheme not only has efficient instantiations but also gets rid of both random oracles and general zero-knowledge proofs. In this paper, we first show that their DCS transformation does not meet the desired security requirements by identifying two security flaws. Then, we point out the reasons that cause those flaws and further propose a secure improvement to fix the flaws. Finally, we present a new generic and efficient DCS scheme without using any public key encryption and prove its security. To the best of our knowledge, this is the first secure DCS scheme that does not require public key encryption.

Keywords: Designated Confirmer Signature, Digital Signature, Fair Exchange.

1 Introduction

As an important cryptographic primitive, digital signatures are employed to achieve the integrity and authenticity of digital documents. In some scenarios, however, the public verifiability of ordinary signatures is not desired, since the signer may wish the recipient of a digital signature could not show the signature to a third party at will. To control the public verifiability, Chaum and van Antwerpen [12] introduced the concept of *undeniable signatures*. Different from ordinary signatures, undeniable signatures cannot be verified without the help of the signer. Naturally, the signer can only confirm valid signatures or disavow invalid signatures.

However, undeniable signatures will not be verifiable if the signer is unavailable or unwilling to help a verifier. To overcome this weakness, *designated confirmer signature* (DCS) schemes were suggested by Chaum at Eurocrypt 1994 [13]. In a DCS scheme, the ability of verifying signatures is delegated to a semi-trusted third party, called *the designated confirmer*. If necessary, the confirmer can further selectively convert individual designated confirmer signatures into ordinary signatures in such a way that anybody can check their validity. In the last decade, a number of DCS schemes [33, 30, 14, 10, 28, 32, 25] have been proposed. However, most of those schemes are either inefficient or insecure.

Okamoto [33] presented the first formal model for DCS and proved that the notion of DCS is in fact equivalent to that of public key encryption. But Michels and Stadler [30] showed that in Okamoto's concrete DCS schemes the confirmer can forge valid signatures on behalf of the signer. Realizing this problem, they further proposed a new security model and constructed efficient DCS schemes secure in their model. However, Camenisch and Michels [10] identified an attack against the DCS schemes proposed in [13, 33, 30] such that the validity of a DCS issued by a signer S can be linked to that of a DCS issued by another signer S'. Therefore, those schemes are insecure if multiple signers share the same confirmer, though this seems to be natural in e-commerce applications, such as fair exchange of digital signatures [1–4], fair e-payment schemes [8, 14], and fair contract signing [22]. Based on this observation, a new model that covers this kind of attacks was proposed in [10]. At the same time, Camenisch and Michels also suggested a generic DCS scheme, which is realized by encrypting an ordinary signature under the confirmer's public key. This construction is provably secure, but inefficient since proving the correctness of such an encryption usually relies on general zero-knowledge proofs for NP statements. In [28], Goldwasser and Waisbard proposed several DCS schemes without appealing to either random oracles [5] or generic zero-knowledge proofs. They achieved this goal by weakening the security requirements of Okamoto [33] and exploiting *strong witness hiding proofs of knowledge*, instead of zero-knowledge proof of knowledge. But their Disavowal protocol (used to disavow an invalid DCS) is still inefficient since it requires general ZK proofs. Monnerat and Vaudenay [32] naturally extended Chaum's DCS scheme [13], but the resulting scheme is provably secure only under non-adaptive chosen-message attack.

At Asiacrypt 2005, Gentry, Molnar and Ramzan [25] presented a generic transformation to convert *any* secure signature scheme into a DCS scheme. Their basic idea is to add "a layer of indirection" in the signature generation procedure. More precisely, in their scheme the signer generates a DCS by issuing an ordinary signature on the commitment of a message and encrypting the randomness used for commitment separately. They proved the DCS scheme constructed in this manner is secure in their security model, which is an enhancement of the model proposed in [28]. Their transformation is interesting, since it gives rise to an efficient and generic DCS scheme without appealing to both random oracles and general zero-knowledge proofs.

In this paper, we first identify two security flaws in Gentry et al's DCS transformation [25] by showing that their scheme does not meet two essential security requirements under their security model. Specifically, we present two attacks against their DCS scheme, in which (a) the confirmer and the signer can collude together to cheat a verifier by issuing a confirmable but invalid signature, and (b) an adaptive attacker can check the validity of a DCS without directly asking for the confirmer's help on this signature. We then point out the reasons causing those flaws and propose an improvement to fix the flaws. Finally, we propose a new generic and efficient DCS scheme without using public key encryption and prove its security. To the best of our knowledge, this is the first generic and secure DCS scheme that does not rely on any public key encryption.

Table 1 gives a brief comparison between our DCS constructions and other existing efficient DCS schemes. Similar to the comparison made in [25], we also compare those DCS schemes in three categories, i.e., whether the scheme relies on the random oracle model [5], which kinds of the basic underlying signatures are used, and how about the computational efficiency. Actually, most items are adopted from [25]. A difference is that in Table 1, we also compare the efficiency of ConfirmedSign protocols in those schemes, which is not discussed in [25]. In Table 1, we list the estimated numbers of exponentiations needed in each interactive protocol. Note that those numbers include the computational overheads introduced by the transformation from a SHVSK protocol to a CZK protocol (See Section 2). In addition, as we shall see in Section 4.2, to achieve the soundness of Confirm protocol both the GW [28] and GMR [25] schemes should be updated. Naturally, this will introduce additional overheads. Due to this reason, an asterisk (*) is marked to the corresponding Confirm protocols. From this comparison, we can see that both of our improved GMR scheme and new DCS scheme have comparable efficiency with the original GMR scheme. Especially, our new DCS scheme without using public key encryption has a very efficient Disavowal protocol, though its security relies on the random oracle. According to our analysis in Section 4.2, however, the GMR scheme suffers two security weaknesses.

Table 1. Comparison of DCS Schemes

	Random Oracle	Underlying Signature	ConfirmedSign Protocol	Confirm Protocol	Disavowal Protocol
CM [10]	Yes	RSA-FDH	-	24λ	60λ
GW [28]	No	CS [17]	5λ	2λ *	generic ZK
GW [28]	No	GMR [27]	5λ	2λ *	generic ZK
GW [28]	No	GHR [23]	5λ	2λ *	generic ZK
GMR [25]	No	Any	20	10 *	41
Improved GMR	No	Any	15	25	60
Our New DCS	yes	Any	15	15	16

The rest of this paper is organized as follows. In Section 2, we introduce notations and some primitives. Section 3 describes the security model of a DCS scheme. Section 4 reviews and analyzes Gentry et al.'s DCS scheme (called GMR scheme for simplicity). We then improve the GMR scheme in Section 5 and propose a new DCS scheme without any public key encryption in Section 6.

2 Preliminaries

Notations. Throughout the paper, λ denotes the security parameter, which is a positive integer. We use $\mathsf{negl}(\lambda)$ to denote a *negligible function* in λ. For a positive integer a, $[a]$ is defined as the set of $\{0, 1, \cdots, a-1\}$. For three integers a, b and c with $c > 0$, $a = b$ rem c denotes the *balanced remainder* of b modulo c. Namely, $a = b + kc \in [-c/2, c/2)$ for some integer k. If $\mathsf{Alg}(\cdot, \cdot, \cdots)$ is a probabilistic algorithm, then $x \leftarrow \mathsf{Alg}(x_1, x_2, \cdots)$ denotes the output x of algorithm Alg on inputs x_1, x_2, $\cdots$, according to the probabilistic distribution determined by Alg's random choices.

Zero-Knowledge Proof. In the setting of DCS schemes, we usually need *concurrent zero-knowledge* (CZK) protocols rather than *special honest-verifier zero-knowledge* (SHVZK) protocols [15]. The reason is that an adversary in DCS schemes may act as an arbitrary cheating verifier during the execution of protocols that confirm or disavow an alleged designated confirmer signature. Briefly speaking, an interactive proof (P, V) for a language L is a CZK protocol if (a) There is a simulator that can simulate transcripts of interaction between Prover P and Verifier V; (b) There is a probabilistic polynomial-time (PPT) knowledge extractor E who can extract a witness (knowledge) given oracle access to Prover P, where E could be rewound if necessary; and (c) Prover P can execute the protocol with one or multiple verifiers in any concurrent way.

Fortunately, there are well known approaches [26, 18, 19, 24] that can efficiently transform SHVZK protocols to CZK protocols. Specifically, Gennaro's approach [24] based on multi-trapdoor commitments has simple structure, while Cramer-Damgård-MacKenzie (CDM) approach [18] can be realized without introducing additional intractability assumptions. In [25], the CDM approach is suggested to use. In our constructions, we would like to select Gennaro's approach due to its simplicity in structure. In any case, the signer or the confirmer (in the GMR scheme and our DCS constructions) will use such CZK protocols to convince a verifier that an alleged message-signature pair is either valid or invalid. However, the verifier (or a number of colluding verifiers) cannot convince the same fact to a third party, even if he/she (or they) executes those verification protocols in any concurrent way as many polynomial times as possible.

CS-Paillier Cryptosystem. An efficient instance of the GMR DCS scheme [25] uses an adaptation of Paillier-based encryption scheme [34] proposed by Camenisch and Shoup in [11]. For simplicity, we call this encryption scheme "CS-Paillier cryptosystem", which can be exploited to realize verifiable encryption of discrete logarithms conveniently. The CCA2 security of this scheme relies on the *decisional composite residuosity assumption* (DCRA) in $\mathbb{Z}^*_{n^2}$, where $n = pq$ is the product of two Sophie-Germain primes p and q (i.e., there exist two primes p' and q' such that $p = 2p' + 1$ and $q = 2q' + 1$). Informally, the DCRA states that it is infeasible to distinguish random elements from $\mathbb{Z}^*_{n^2}$ and random elements from the subgroup consisting of all n-th powers of elements in $\mathbb{Z}^*_{n^2}$.

Now, we briefly review this encryption scheme (refer to [11] for details). The user generates a composite modulus $n = pq$ as above. The user's public key includes a collision-resistant hash function H, $h = 1 + n$, a random $g' \in \mathbb{Z}^*_{n^2}$, and values $g = {g'}^{2n}, y_1 = g^{x_1}, y_2 = g^{x_2}$, and $y_3 = g^{x_3}$, where $x_1, x_2, x_3 \in_R [n^2/4]$ constitute the private key. Define a function $\mathsf{abs}(\cdot) : \mathbb{Z}_{n^2} \to \mathbb{Z}_{n^2}$ as $\mathsf{abs}(a) = a$ if $0 \le a \le n^2/2$, or $\mathsf{abs}(a) = n^2 - a \bmod n^2$ if $n^2/2 < a < n^2$.

To encrypt a value $r \in [n]$ with a label $L \in \{0,1\}^*$, the sender picks $t \in_R [n/4]$ and computes a triple (u, e, v) by $u = g^t$, $e = y_1^t h^r$, and $v = \mathsf{abs}((y_2 y_3^{H(u,e,L)})^t)$. The resulting ciphertext (u, e, v) with label L can be decrypted follows. First, the user checks whether $\mathsf{abs}(v) \equiv v$ and $u^{2(x_2 + H(u,e,L)\cdot x_3)} \equiv v^2$. If any check fails, output $\perp$. Otherwise, the user computes $\hat{r} = (e/u^{x_1})^{2k}$ for $k = 2^{-1} \bmod n$. If $\hat{r}$ is of form h^r for some $r \in [n]$ (i.e., $\hat{r} - 1$ is divisible by n), then output $r = (\hat{r} - 1)/n \in [n]$. Otherwise, output $\perp$.

3 Security Model and Definitions of DCS

We now review the security model and definitions of designated confirmer signatures (DCS) following Gentry et al.'s exposition in [25]. Specifically, the syntax of DCS is the same as given in [25], while the security definitions are improved in some minor ways mainly for readability. A DCS scheme has three different roles of parties: a signer S, a verifier V, and a designated confirmer C.

Definition 1 (Syntax). A `designated confirmer signature (DCS)` scheme consists of a tuple of probabilistic polynomial-time (PPT) algorithms and interactive protocols, ($\mathsf{DCGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{ConfirmedSign}_{(S,V)}, \mathsf{Confirm}_{(C,V)}, \mathsf{Disavowal}_{(C,V)}$), as described below.

– DCGen: As the key generation algorithm of DCS, it takes as input the security parameter 1^λ, and outputs two pairs of keys (sk_S, pk_S) and (sk_C, pk_C). Here, (sk_S, pk_S) are the signer S's signing and verification keys respectively, while (sk_C, pk_C) the confirmer C's private and public keys respectively [3].
– Sign: takes as input a message m and a signing key sk_S, and outputs a basic signature σ such that $\mathsf{Verify}(m, \sigma, pk_S) = \mathsf{Accept}$.
– Verify: takes as input a triple (m, σ, pk_S), and outputs Accept if σ is an output of $\mathsf{Sign}(m, sk_S)$ *or* $\perp$ otherwise.
– Extract: takes as input (m, σ', sk_C, pk_S), and outputs a string σ such that $\mathsf{Verify}(m, \sigma, pk_S) = \mathsf{Accept}$ if σ is an output of $\mathsf{Sign}(m, sk_S)$, or $\perp$ otherwise.

[3] As pointed in [25], for simplicity DCGen is here denoted as a single algorithm. In a real implementation, the signer S and confirmer C would generate their key pairs separately, using two distinct algorithms SGen and CGen, so that C does not learn sk_S and S does not learn sk_C.

– $\mathsf{ConfirmedSign}_{(S,V)}$: an interactive protocol between the signer S (with private input sk_S) and a verifier V with common input (m, pk_S, pk_C). The output of V is a pair (b, σ') where $b \in \{\mathsf{Accept}, \perp\}$ and σ' is S's designated confirmer signature on message m. For some verifier V, the $\mathsf{ConfirmedSign}$ protocol should be complete and sound.
  - **Completeness**: There is some signer S such that for any (valid) signer and confirmer keys, any message m, the $\mathsf{ConfirmedSign}$ protocol outputs $(\mathsf{Accept}, \sigma')$, where $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', sk_C, pk_S), pk_S) = \mathsf{Accept}$.
  - **Soundness**: For any signer S', if $\mathsf{ConfirmedSign}_{(S',V)}(m, pk_S, pk_C) = (\mathsf{Accept}, \sigma')$, then

$$\Pr[\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', sk_C, pk_S), pk_S) = \perp] < \mathsf{negl}(\lambda). \quad (1)$$

    In other words, even a cheating signer S' cannot convince an honest verifier V that an "un-extractable" DCS σ' is valid.

– $\mathsf{Confirm}_{(C,V)}$: an interactive protocol between the confirmer C and a verifier V to confirm a valid DCS σ'. The common input is (m, σ', pk_S, pk_C), and C's private input is sk_C, while the output is $b \in \{\mathsf{Accept}, \perp\}$. For some verifier V, the $\mathsf{Confirm}$ protocol must be both complete and sound.
  - **Completeness**: There is some C such that if $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', sk_C, pk_S), pk_S) = \mathsf{Accept}$ then $\mathsf{Confirm}_{(C,V)}(m, \sigma', pk_S, pk_C) = \mathsf{Accept}$.
  - **Soundness**: For any confirmer C', if $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', sk_C, pk_S), pk_S) = \perp$, then

$$\Pr[\mathsf{Confirm}_{(C',V)}(m, \sigma', pk_S, pk_C) = \mathsf{Accept}] < \mathsf{negl}(\lambda). \quad (2)$$

    That is, even a cheating confirmer C' cannot convince an honest verifier V that an "un-extractable" DCS σ' is valid.

– $\mathsf{Disavowal}_{(C,V)}$: an interactive protocol between the confirmer C and a verifier V to disavow an invalid DCS σ'. Given the common input (m, σ', pk_S, pk_C) and C's private input sk_C, the $\mathsf{Disavowal}$ protocol outputs $b \in \{\mathsf{Accept}, \perp\}$. For some verifier V, the protocol must be complete and sound.
  - **Completeness**: There is a confirmer C such that if $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', sk_C, pk_S), pk_S) = \perp$, then $\mathsf{Disavowal}_{(C,V)}(m, \sigma', pk_S, pk_C) = \mathsf{Accept}$.
  - **Soundness**: For any PPT confirmer C', if $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', sk_C, pk_S), pk_S) = \mathsf{Accept}$, then

$$\Pr[\mathsf{Disavowal}_{(C',V)}(m, \sigma', pk_S, pk_C) = \mathsf{Accept}] < \mathsf{negl}(\lambda). \quad (3)$$

    In other words, even a cheating confirmer C' cannot convince an honest verifier V that an "extractable" DCS σ' is invalid. □

We consider three security requirements of a designated confirmer signature scheme, each of which is from the view point of a different role in a DCS scheme. More specifically, a DCS should be: (a) *secure for verifiers*, i.e., confirmed DCS should be extractable and disavowed DCS should be un-extractable; (b) *secure for the signer*, i.e., anybody else (including the confirmer) should be unable to forge a DCS on a new message unsigned by the signer; and (c) *secure for the confirmer*, i.e., only the confirmer can confirm or disavow an alleged DCS.

For the purposes of the security model, a two-move protocol $\mathsf{OutputDCS}_{(S,V)}$ is also introduced, which is the stunted version of $\mathsf{ConfirmedSign}_{(S,V)}$ in which V queries m and S outputs a DCS σ' on m without confirming its correctness. In the following, the adversary $\mathcal{A}$ is allowed to access a collection of oracles $\mathcal{O} = \{\mathsf{ConfirmedSign}_{(S,\mathcal{A})}, \mathsf{Confirm}_{(C,\mathcal{A})}, \mathsf{Disavowal}_{(C,\mathcal{A})}, \mathsf{Extract}\}$ for: 1) receiving a confirmed signature on a message of its choice (via the $\mathsf{ConfirmedSign}_{(S,\mathcal{A})}$ oracle); 2) executing the interactive $\mathsf{Confirm}_{(C,\mathcal{A})}$ protocol in the verifier role; 3) executing the interactive protocol $\mathsf{Disavowal}_{(C,\mathcal{A})}$ in the verifier role; and 4) getting a basic signature from a designated confirmer signature via the $\mathsf{Extract}$ oracle. Furthermore, since we consider the security of a DCS scheme with multiple signers, any adversary in the following definitions is allowed at any time to generate additional signature pairs $(sk_{S'}, pk_{S'})$ (not necessary by running the key generation algorithm DCGen) and to interact with the confirmer C with respect to those keys.

Informally, security for verifiers requires that even if the adversary $\mathcal{A}$ compromises the private keys of both the confirmer C and the signer S simultaneously, it is still unable to create a pair (m, σ') that will be confirmed (via either $\mathsf{ConfirmedSign}$ or $\mathsf{Confirm}$) even though (m, σ') is un-extractable, or that will be disavowed (via $\mathsf{Disavowal}$) even though (m, σ') is extractable. For simplicity, in the following descriptions we use π to denote the public parameters $(1^\lambda, pk_S, pk_C)$.

Definition 2 (**Unfoolability: Security for Verifiers**). Formally, we say a DCS scheme is *secure for verifiers* if for any PPT algorithm $\mathcal{A}$ involved in the experiment `Exp1-UnFoolVerifier`, its advantage $\mathsf{Adv}^{\mathsf{fool}}(\mathcal{A}) := \Pr[b_{\mathsf{fool}} = 1] < \mathsf{negl}(\lambda)$, where b_{fool} is the one bit information returned by the experiment.

```
Exp1-UnfoolVerifier:
1. $(sk_S, pk_S, sk_C, pk_C) \leftarrow \mathsf{DCGen}(1^\lambda)$
2. $(m, \sigma', \tau_1, \tau_2, \tau_3) \leftarrow \mathcal{A}_0^{\mathcal{O}}(sk_S, sk_C, \pi)$
3. $(b_1, \sigma') \leftarrow \mathsf{ConfirmedSign}_{(\mathcal{A}_1(\tau_1),V)}(m, \pi)$ in Case 1
4. $b_2 \leftarrow \mathsf{Confirm}_{(\mathcal{A}_2(\tau_2),V)}(m, \sigma', \pi)$ in Case 1
5. $b_3 \leftarrow \mathsf{Disavowal}_{(\mathcal{A}_3(\tau_3),V)}(m, \sigma', \pi)$ in Case 2
6. Return $b_{\mathsf{fool}} = (b_1 = \mathsf{Accept} \vee b_2 = \mathsf{Accept} \vee b_3 = \mathsf{Accept})$.
```

Note that here "Case 1" and "Case 2" refer to the restraint conditions on the adversary's output (m, σ'):

– Case 1: $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', sk_C, pk_S), pk_S) = \perp$, i.e., σ' is un-extractable.
– Case 2: $\mathsf{Verify}(m, \mathsf{Extract}(m, \sigma', sk_C, pk_S), pk_S) = \mathsf{Accept}$, i.e., σ' is extractable.
□

Security for the signer informally requires that an adversary $\mathcal{A}$ (including the confirmer C) must be unable to forge a valid DCS pair (m, σ') for a new message m, though it may be able to create an extractable or confirmable (via either $\mathsf{ConfirmedSign}$ or $\mathsf{Confirm}$) (m, σ'') for a signed message m.

For each DCS scheme, we can specify an efficiently computable equivalence relation R, and say (m, σ') and (m, σ'') are *equivalent* if and only if $R(m, \sigma', \sigma'') = 1$. For example, if a DCS scheme is assumed to be *strongly existentially unforgeable*, it may be appropriate to define $R(m, \sigma', \sigma'') = 1$ iff $\sigma' = \sigma''$. However, the relation R depending on the concrete implementation may need not be so restrictive.

Definition 3 (**Unforgeability: Security for the Signer**). We formally say a DCS scheme is *secure for the signer* if for any PPT adversary $\mathcal{A}$ involved in the following experiment `Exp2-UnForge`, its advantage $\mathsf{Adv}^{\mathsf{forge}}(\mathcal{A}) := \Pr[b_{\mathsf{forge}} = 1] < \mathsf{negl}(\lambda)$, where b_{forge} is the one bit information returned by the experiment. Note that in the experiment, L_{sig} denotes the list of all message-signature pairs (m_i, σ'_i) output by the $\mathsf{ConfirmedSign}$ oracle in Step 2 and all (m_i, σ''_i) such that $R(m_i, \sigma'_i, \sigma''_i) = 1$.

```
Exp2-UnForge:
1. $(sk_S, pk_S, sk_C, pk_C) \leftarrow \mathsf{DCGen}(1^\lambda)$
2. $(m, \sigma') \leftarrow \mathcal{A}^{\mathcal{O}}(\pi, sk_C)$
3. $b \leftarrow \mathsf{Verify}(m, \sigma, pk_S)$ for $\sigma = \mathsf{Extract}(m, \sigma', sk_C, pk_S)$
4. Return $b_{\mathsf{forge}} = (b = \mathsf{Accept} \wedge (m, \sigma') \notin L_{sig})$.
```
□

Security for the confirmer informally requires that the evidences of confirmation or disavowal of a DCS σ' should be *non-transferable*. Namely, the transcript of a proof of knowledge in $\mathsf{Confirm}_{(C,V_1)}(m, \sigma', pk_S, pk_C)$ or $\mathsf{ConfirmedSign}_{(C,V_1)}(m, \sigma', pk_S, pk_C)$ should not convince V_2 ($\neq V_1$) that σ' signs m, while the transcript of a proof of knowledge in $\mathsf{Disavowal}_{(C,V_1)}(m, \sigma', pk_S, pk_C)$ should not convince V_2 ($\neq V_1$) that σ' does not sign m. To guarantee that a DCS scheme satisfies non-transferability, i.e., the transcripts in those protocols are unconvincing, we require that those transcripts be simulatable. In a DCS scheme with non-transferability, even if verifier V_1 already knew the validity of a message-signature pair (m, σ') (via interacting with the signer or the confirmer), it cannot convince verifier V_2 to believe this fact, since all the evidences provided by V_1 could be simulated, i.e., not true transcripts from real executions of the $\mathsf{ConfirmedSign}$, $\mathsf{Confirm}$ or $\mathsf{Disavowal}$ protocols.

In the following formal definition, algorithms $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}'_1$ represent verifier V_1, verifier V_2 and a simulation algorithm, respectively. If $\mathcal{A}_2$ has only negligible advantage to guess whether its input τ came from $\mathcal{A}_1$ or $\mathcal{A}'_1$, this suggests that $\mathcal{A}_1$'s potentially authentic transcript showing that m_0 was signed is no more convincing or informative than $\mathcal{A}'_1$'s simulated transcript (falsely) showing that m_1 was signed. In the security proof, $\mathcal{A}'_1$ will use $\mathcal{A}_1$ as a subroutine, and will simulate correct answers to $\mathcal{A}_1$'s oracle queries.

Definition 4 (**Transcript Simulatability: Security for the Confirmer**). Formally, we say a DCS scheme is *secure for the confirmer* if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ involved in the following experiment `Exp3-Transcript Simulatability`, there exists a PPT algorithm $\mathcal{A}'_1$ such that $\mathcal{A}$'s advantage respect to $\mathcal{A}'_1$ is negligible in the security parameter. Namely, $\mathsf{Adv}^{\mathsf{trans}}(\mathcal{A}, \mathcal{A}'_1) := |\Pr[b_{\mathsf{trans}} = 1] - 1/2| < \mathsf{negl}(\lambda)$, where b_{trans} is the one bit information returned by the experiment. In the experiment, $\mathcal{A}_0$ with sk_S first outputs two messages m_0 and m_1 and some state s. Then, a DCS σ' on m_0 or m_1 is output randomly by ConfirmedSign. After that, $\mathcal{A}_1$, $\mathcal{A}'_1$ and $\mathcal{A}_2$ play a game in which $\mathcal{A}'_1$ tries to make its output (when m_1 is signed) look indistinguishable from $\mathcal{A}_1$'s output (when m_0 is signed); $\mathcal{A}_2$ attempts to distinguish whether its input τ came from $\mathcal{A}_1$ or $\mathcal{A}'_1$. In the experiment, $\mathcal{A}_1$ gets oracle accesses $\mathcal{O}_1$, i.e., all oracles in $\mathcal{O}$ under the restriction that $(m_0, \sigma'), (m_1, \sigma') \notin L_{ext}$, where L_{ext} is a list consisting of 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 $R(m_i, \sigma'_i, \sigma''_i) = 1$ [4]. On the other hand, $\mathcal{A}'_1$ is given very limited oracle accesses, i.e., it can make only q OutputDCS queries as long as $\mathcal{A}_1$ makes at most q ConfirmedSign queries. $\mathcal{A}_2$ is given access to oracles in $\mathcal{O}_2$, i.e., all oracles in $\mathcal{O}$ with the restriction that $\mathcal{A}_2$ cannot make any oracle 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, the distinguishing task of $\mathcal{A}_2$ will become trivial.). Finally, $\mathcal{A}_2$ outputs one bit information b' as its guess to the value of b, i.e., whether m_0 or m_1 is signed.

```
Exp3-TranscriptSimulatability:
1. $(sk_S, pk_S, sk_C, pk_C) \leftarrow \mathsf{DCGen}(1^\lambda)$
2. $(m_0, m_1, s) \leftarrow \mathcal{A}_0^{\mathcal{O}}(\pi, sk_S)$
3. $b \leftarrow_R \{0, 1\}$
4. $(\mathsf{Accept}, \sigma') \leftarrow \mathsf{ConfirmedSign}(\pi, m_b)$
5. If $b = 0$, $\tau \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(\pi, b, m_0, m_1, s, \sigma')$;
   else,      $\tau \leftarrow \mathcal{A}_1'^{\mathsf{OutputDCS}}(\pi, b, m_0, m_1, s, \sigma')$
6 $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(\pi, m_0, m_1, \tau, \sigma')$
7. Return $b_{\mathsf{trans}} = ((b' = b) \wedge ((m_0, \sigma') \notin L_{ext}) \wedge ((m_1, \sigma') \notin L_{ext}))$.   □
```

Note that the above transcript simulatability is not the strongest requirement, but it is strong enough, as pointed out in [25] and further explained below. On the one hand, the transcript is not *perfectly* simulatable in the sense that σ' may convince verifier $\mathcal{A}_2$ that the signer indeed signed some message m, though $\mathcal{A}_2$ cannot tell which specific message (i.e. m_0 or m_1) was signed. So this security requirement is weaker than that given in [33]. On the other hand, the above security model actually prevents *confirmer impersonation*. Namely, even if an adversary $\mathcal{B}$ controls sk_S it cannot impersonate the confirmer by executing Extract, Confirm, Disavowal, or ConfirmedSign associated to a pair $(m, \sigma') \in L_{sig} \backslash L_{ext}$ (See the discussions provided in [25]).

Moreover, we would like to point out that the above transcript simulatability also implies the property of *invisibility*, whose formal definition is given by Camenisch and Michels in [10] [5]. Informally, invisibility requires that an adaptively chosen message attacker $\mathcal{A}$ cannot correctly guess a newly issued DCS σ' is for m_0 or m_1 with probability better than 1/2 non-negligibly. It is not difficult to see that the corresponding experiment for invisibility can be obtained

[4] Otherwise, $\mathcal{A}_1$ could trivially give $\mathcal{A}_2$ explicit proof that m_0 was signed by revealing the extraction of σ'.

[5] Galbraith and Mao [21] formally specified another definition of invisibility, which is a little stronger than the version given in [10]. For many real life applications, however, it seems (weak) invisibility is enough.

from `Exp3-TranscriptSimulatability` by deleting algorithms $\mathcal{A}_1$ and $\mathcal{A}_1'$ (i.e. Step 5), and deleting τ in the input of algorithm $\mathcal{A}_2$. From this observation, we know that invisibility is implied by transcript simulatability. However, note that according to the result in [10], the DCS schemes in [13, 33, 30] *do not* satisfy invisibility.

Definition 5 (**Security of a DCS Scheme**). We say a designated confirmer signature scheme is `secure`, if it satisfies security for verifiers, the signer, and the confirmer. That is, the DCS scheme meets the formal requirements given in definitions 2, 3 and 4 simultaneously. □

4 The GMR Scheme and Its Security

4.1 Review of the GMR Scheme

This section reviews the GMR scheme [25], which is a generic construction that transforms any existentially unforgeable signature scheme [27] into a DCS scheme. In this transformation, the following two primitives are required: an IND-CCA2 secure encryption scheme PKE [20], and a statistically hiding computationally binding commitment scheme $\mathsf{Com}(m, r)$. The basic idea is to issue a DCS on a message m the signer signs on a commitment $\mathsf{Com}(m, r)$ instead of m itself. To guarantee that the confirmer can open a commitment $\mathsf{Com}(m, r)$ with respect to m, the randomness r used in commitment is encrypted under the confirmer's public key and the resulting ciphertext c is attached as a component of DCS. To prove the validity of such a DCS, the signer or the confirmer convinces a verifier that ciphertext c is properly prepared by exploiting zero knowledge proofs secure against cheating verifiers. The following is the high-level descriptions of the GMR scheme.

- DCGen: The signer S uses a secure digital signature scheme DSS=(SGen, Sig, Ver), and creates a key pair $(sk_S, pk_S) \leftarrow \mathsf{SGen}(1^\lambda)$. The confirmer C uses an IND-CCA2 encryption scheme PKE=(CGen, Enc, Dec), and creates key pair $(sk_C, pk_C) \leftarrow \mathsf{CGen}(1^\lambda)$. Note that 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, the signer S creates a statistically hiding and computationally binding commitment $\psi = \mathsf{Com}(m, r)$ to the message m by selecting randomness r and creates $\sigma^* = \mathsf{Sig}((\psi, c, pk_S), sk_S)$. The basic signature is $\sigma = (\sigma^*, c, r)$.
- $\mathsf{Extract}$: On input $\sigma' = (\sigma^*, \psi, c)$ and m, it outputs r if $\sigma^* = \mathsf{Sig}((\psi, c, pk_S), sk_S)$ and the confirmer C can derive $r = \mathsf{Dec}(sk_C, c)$ so that $\psi = \mathsf{Com}(m, r)$. Otherwise, it outputs $\perp$.
- $\mathsf{ConfirmedSign}$: In addition to the above steps in the Sign procedure, the signer S also computes the ciphertext $c = \mathsf{Enc}(pk_C, r)$. The designated confirmer signature is $\sigma' = (\sigma^*, \psi, c)$, where $\sigma^* = \mathsf{Sig}((\psi, c, pk_S), sk_S)$. The signer also performs a ZK proof of knowledge of a value r such that $\psi = \mathsf{Com}(m, r)$ and $c = \mathsf{Enc}(pk_C, r)$.

– $\mathsf{Confirm}$: The confirmer C first checks that (ψ, c, pk_S) has been signed with sk_S using the provided pk_S, and aborts if the check fails. Then, C performs a ZK proof of knowledge of a value r such that $\psi = \mathsf{Com}(m, r)$.
– $\mathsf{Disavowal}$: To disavow a purported signature $\sigma' = (\sigma^*, \psi, c)$ on message m, the confirmer C does the following. C first checks if c is a valid encryption of some r. If not, it performs a ZK proof of knowledge that the string c is not a well-formed encryption. Otherwise, C computes $r' = \mathsf{Dec}(sk_C, c)$. If $\psi \neq \mathsf{Com}(m, r')$, then C provides a ZK proof of knowledge that there is a value r' such that $\psi \neq \mathsf{Com}(m, r')$ and $r' = \mathsf{Dec}(sk_C, c)$.

Gentry et al. pointed out that all the above statements involving ZK proofs can be expressed as NP statements (with short witnesses). Therefore, in theory the above generic DCS scheme can be implemented in polynomial time from any suitably secure encryption scheme, commitment scheme, and signature scheme. Since generic ZK proofs for NP-statements are not very practical, they suggested that an efficient instantiation can be obtained by selecting the CS-Paillier encryption scheme [11], and the Pedersen commitment scheme [35] over a prime order group Γ. However, in [25] this instantiation was just given in a high-level description without implementation details.

4.2 Security of the GMR Scheme

The GMR scheme reviewed above is an interesting designated signature scheme, since it is a generic transformation with efficient implementations. In this section, however, we shall identify two security flaws in the GMR scheme. Namely, it *does not* satisfy the unfoolablility and the invisibility.

According to the definition of security for verifiers, an adversary $\mathcal{A}$ should not be able to create confirmable but un-extractable DCS message-signature pair (m, σ') or disavowable but extractable pair (m, σ'), even if the adversary $\mathcal{A}$ compromises both sk_S and sk_C, i.e. the private keys of the confirmer S and the signer C. However, according to the specification of the GMR scheme, such an adversary $\mathcal{A}$ can fool a verifier as follows. $\mathcal{A}$ first picks two random numbers r and c (with proper lengths), then computes $\psi = \mathsf{Com}(m, r)$ for an arbitrary message m and issues $\sigma^* = \mathsf{Sig}((\psi, c, pk_S), sk_S)$ using sk_S. The resulting DCS message-signature pair is (m, σ'), where $\sigma' = (\sigma^*, \psi, c)$. Note that σ' can be confirmed by running the $\mathsf{Confirm}$ protocol, since σ^* is S's valid signature on (ψ, c, pk_S) and the adversary $\mathcal{A}$ with the randomness r can provide a ZK proof of knowledge showing that there is a value r such that $\psi = \mathsf{Com}(m, r)$. In the experiment `UnFoolVerifier`, this attack allows V to output $b_2 = 1$ with probability 1.

In their security claim (Theorem 1 in [25]), Gentry et al. pointed out that security for verifiers follows the soundness of $\mathsf{ConfirmedSign}$, $\mathsf{Confirm}$ and $\mathsf{Disavowal}$ protocols. This is a correct reasoning, but in the context of their DCS scheme the assumption is not true. Because their $\mathsf{Confirm}$ protocol is actually not sound, as demonstrated by the above attack. Based on this observation, we can simply get a sound $\mathsf{Confirm}$ protocol by requiring the confirmer C to prove in ZK that it knows a value r such that $\psi = \mathsf{Com}(m, r)$ and $r = \mathsf{Dec}(sk_C, c)$.

Remark 1: Note that a similar attack applies to the GW generic DCS scheme proposed by Goldwasser and Waisbard in [28], since their Confirmation Protocol *does not* satisfy the soundness too (For detail, please check the first paragraph of page 91 [28]). That is, a verifier can be fooled by an un-extractable DCS signature, if the signer and the confirmer collude together or their private keys are compromised by an adversary. Naturally, we could repair the GW scheme as suggested above. By doing so, however, the resulting Confirmation Protocol will become less efficient than the original one.

Informally, *invisibility* means that an (adaptively chosen message) adversary $\mathcal{A}$ cannot distinguish whether a DCS σ' is on message m_0 or message m_1 with non-negligible advantage better than 1/2, even if $\mathcal{A}$ is given the signer's private key sk_S. The formal definition of this version of invisibility can be found in [10]. Now, we present an attack that breaches invisibility and then breaks transcript simulatability, since the latter implies the former, as we mentioned before.

Our attack is based on the observation that in the GMR DCS scheme, the ciphertext c of randomness r could be re-used in different signatures. For simplicity, let us demonstrate the attack in the scenario where Pedersen commitment scheme [35] and Cramer-Shoup CCA2 secure encryption scheme [16] are used in the GMR scheme. That is, we compute $\psi = \mathsf{Com}(m, r) := \delta^m \gamma^r$, where δ and γ are two random generators of a group Γ with prime order ρ. Let $\sigma' = (\sigma^*, \psi, c)$ be a valid DCS for message m_0 or message m_1 with exact probability 1/2. So, there exists $b \in \{0, 1\}$ and a value r such that $\sigma^* = \mathsf{Sig}((\psi, c, pk_S), sk_S)$, $\psi = \delta^{m_b}\gamma^r$, and $c = \mathsf{Enc}(pk_C, r)$. The goal of an adversary $\mathcal{A}$ is to tell whether the bit b equals 1 or 0. To this end, the adversary $\mathcal{A}$ first picks an arbitrary message m' ($m' \neq m_0$ and $m' \neq m_1$), and computes $\psi' = \psi\delta^{m'}\delta^{-m_0}$ $(= \delta^{m'+m_b-m_0}\gamma^r)$. Then, $\mathcal{A}$ asks the signing oracle of the underlying signature scheme to get a signature σ^{**} on (ψ', c, pk_S), i.e., $\sigma^{**} = \mathsf{Sig}((\psi', c, pk_S), sk_S)$. After that, $\mathcal{A}$ asks the $\mathsf{Extract}$ oracle by enquiring $(m', \sigma'' = (\sigma^{**}, \psi', c))$. Finally, $\mathcal{A}$ outputs $b = 0$ if a value r is received from the $\mathsf{Extract}$ oracle; otherwise (i.e., the $\mathsf{Extract}$ oracle reveals $\perp$), $\mathcal{A}$ outputs $b = 1$. Note that to correctly guess the random bit b, $\mathcal{A}$ can alternatively run $\mathsf{Confirm}$ or $\mathsf{Disavowal}$ protocol on the same pair (m', σ'') with the confirmer. It is not difficult to see that $\mathcal{A}$ wins the above game with probability 1.

Actually, in the setting of multiple signers, the above adversary $\mathcal{A}$ can also check the validity of signer S's DCS $\sigma' = (\sigma^*, \psi, c)$ on message m by interacting with the confirmer C on another message-signature pair (m', σ'') from signer S''s (different from S). The reason is that $\mathcal{A}$ can collude with S' so that S' issues his DCS σ'' on a new message m' by re-using c similarly, i.e., $\sigma^{**} = \mathsf{Sig}((\psi', c, pk_{S'}), sk_{S'})$, where $\psi' = \psi\delta^{m'}\delta^{-m}$.

In the scenarios of fair exchange, the above attacks may allow one party to cheat the other. In addition, the above attack also implies that the signer is *coercible* [10, 32]. That is, even if the signer S erases the intermediate results (i.e. randomness r etc.) after the computation of a DCS σ', S may still be coerced since a third party can prove the fact that S indeed issued σ'.

5 Improved GMR Scheme

To enhance the transcript simulatability of the GMR scheme, we should let the confirmer know the "context" of the ciphertext c meaning that c is created with respect to which message m and which verification key pk_S. We notice that this can be achieved if the underlying IND-CCA2 secure encryption scheme supports the use of labels. Namely, we can define a label $L = m||pk_S$ so that the confirmer is aware of the context of c. In the following, we describe our improvement on the GMR DCS scheme in the setting of exploiting CS-Paillier encryption scheme [11] and Pedersen commitment [35]. Such a treatment could be helpful to readers who want to know (and apply) a concrete DCS implementation with clearly technical details. At the same time, note that this concrete DCS scheme can be straightforwardly generalized by using any IND-CCA2 secure encryption with labels and any perfectly hiding and computationally binding commitment.

To obtain a verifiable encryption scheme from the CS-Paillier cryptosystem, we assume that there is an additional composite modulus $n_2 = p_2q_2$, where $p_2 = 2p_2'+1$ and $q_2 = 2q_2'+1$ are two safe primes, along with elements $g_2, h_2 \in \mathbb{Z}_{n_2}^*$ of order $p_2'q_2'$. In addition, we select a third group Γ of prime order ρ, along with two random generators δ and γ. In the group Γ, the discrete logarithm problem is assumed to be hard. In our DCS scheme, a message digest m (the hashed value of a real message) shall be committed by $\mathsf{Com}(m, r) = \delta^m\gamma^r$, where $r \in_R [\rho]$. We require $n_2 \neq n$, $\rho = |\Gamma| < n{\cdot}2^{-k-k'-3}$, and $2^k < \min\{p', q', p_2', q_2'\}$ for two further security parameters k and k'. Actually, $\{0,1\}^k$ defines the "challenge space" of the verifier V, while k' controls the quality of the ZK property [11]. In addition, it is required that the prover (a signer S or the confirmer C) does not know the factorization of n_2. So, for simplicity, we just assume that $(n_2, g_2, h_2, \Gamma, \gamma, \delta)$ are generated by a trusted party and viewed as a common reference string.

- DCGen: The signer S generates a key pair $(sk_S, pk_S) \leftarrow \mathsf{SGen}(1^\lambda)$ for any secure digital signature scheme DSS=(SGen, Sig, Ver). The confirmer C generates a key pair $(sk_C, pk_C) \leftarrow \mathsf{CGen}(1^\lambda)$ for the CS-Paillier encryption scheme. Namely, we assume $sk_C = (x_1, x_2, x_3)$ and $pk_C = (n, g, h, y_1, y_2, y_3, H)$.
- Sign: To sign a message $m \in [\rho]$, the signer S first selects a random number $r \in_R [\rho]$, then computes $\psi = \mathsf{Com}(m, r) = \delta^m\gamma^r$ and $\sigma^* = \mathsf{Sig}((\psi, pk_S), sk_S)$. The basic signature for message m is $\sigma = (\sigma^*, r)$.
- Verify: On input an extracted DCS signature $\sigma = (\sigma^*, r)$ for a message m, it returns the output of $\mathsf{Ver}((\psi, pk_S), \sigma^*, pk_S)$, where $\psi = \mathsf{Com}(m, r)$.
- $\mathsf{Extract}$: On input $\sigma' = (\sigma^*, \psi, c)$ and message m, it outputs r if $\sigma^* = \mathsf{Sig}((\psi, pk_S), sk_S)$ and the confirmer C can derive $r = \mathsf{Dec}(sk_C, c)$ w.r.t. label $L = m||pk_S$ such that $\psi = \mathsf{Com}(m, r)$. Otherwise, it outputs $\perp$.
- $\mathsf{ConfirmedSign}$: In addition to the above steps in the Sign procedure, the signer S also computes the ciphertext $c := (u, e, v) = \mathsf{Enc}(pk_C, r)$ under the label $L = m||pk_S$ (recall Section 2). The designated confirmer signature is $\sigma' = (\sigma^*, \psi, c)$, where $\sigma^* = \mathsf{Sig}((\psi, pk_S), sk_S)$. Then, the signer runs a CZK protocol with a verifier to show that c and ψ are properly prepared. That is, the signer provides the following ZK proof of knowledge of values (t, r, s),

where $s \in_R [n_2/4]$ and $\alpha = \psi\delta^{-m}$:

$$PK\{(t,r,s) : u^2 = g^{2t} \wedge e^2 = y_1^{2t}h^{2r} \wedge v^2 = (y_2 y_3^{H(u,e,L)})^{2t} \wedge \alpha = \gamma^r \wedge \ell = g_2^r h_2^s \wedge -n/2 < r < n/2\}. \quad (4)$$

– Confirm: Upon receiving a message-signature pair $(m, \sigma' = (\sigma^*, \psi, c))$ with respect to pk_S, the confirmer C first checks whether σ^* is S's signature on (ψ, pk_S). C aborts if the check fails. Otherwise, C decrypts $c = (u, e, v)$ using label $L = m||pk_S$ to get a value r, and then checks that $\psi \equiv \mathsf{Com}(m, r)$. If any step of this procedure fails, C performs the Disavowal protocol. Otherwise, C needs to show that there is such an r in ZK. That is, the signer provides the following ZK proof of knowledge of values (x_1, x_2, x_3, r, s), where $s \in_R [n_2/4]$ and $\alpha = \psi\delta^{-m}$:

$$PK\{(x_1,x_2,x_3,r,s) : y_1 = g^{x_1} \wedge y_2 = g^{x_2} \wedge y_3 = g^{x_3} \wedge e^2 = u^{2x_1}h^{2r} \wedge v^2 = u^{2x_2}u^{2H(u,e,L)x_3} \wedge \alpha = \gamma^r \wedge \ell = g_2^r h_2^s \wedge -n/2 < r < n/2\}. \quad (5)$$

– Disavowal: To disavow a purported signature $\sigma' = (\sigma^*, \psi, c)$ on message m, the confirmer C does the following. C first checks if c is a valid encryption of some r. If not, it performs a ZK proof of knowledge that the string c is not well-formed. Otherwise, C computes $r = \mathsf{Dec}(sk_C, c)$ and proves in ZK that $\psi \neq \mathsf{Com}(m, r)$. That is, the confirmer C provides a ZK proof for the following statement:

$$[c \text{ is invalid w.r.t. } L = m||pk_S] \text{ OR } [\exists\ r \text{ s.t. } r = \mathsf{Dec}(sk_C, c) \text{ AND } \psi \neq \mathsf{Com}(m, r)]. \quad (6)$$

Compared with the original GMR scheme, there are three main changes in the above improvement. First, our basic signature is a pair (σ^*, r) instead of a triple (σ^*, c, r) in GMR scheme, where c is treated as auxiliary information in [25]. Our proposal not only becomes simpler, but also avoids the potential question whether a proof should be provided to show that c indeed encrypts r. We also remark that the algorithm Verify is not specified in the GMR scheme [25]. Second, the Confirm protocol is enhanced to guarantee the soundness, as we mentioned before. Third, we explicitly specify how to use labels in the DCS scheme. In contrast, the authors of [25] claimed that *any* IND-CCA2 secure encryption scheme [20] can be used by the confirmer without mentioning how to use the labels in their instantiation, where CS-Paillier cryptosysem is exploited.

In addition, a practical implementation should guarantee that the Extract algorithm is performed correctly. To this end, we can require that the confirmer first runs Confirm or Disavowal protocol with a verifier, and then outputs a correct value r or $\perp$ respectively. Alternatively, the confirmer can provide some non-interactive proof to show that it did this properly. For example, the confirmer can perform the non-interactive version of Confirm or Disavowal protocol with additional output r or $\perp$ correspondingly.

The implementation details and the security proof of the improved GMR scheme can be found in the full version [37]. The following theorem summarizes the security result on this DCS scheme.

Theorem 1. *Let DSS = (SGen, Sig, Ver) be any signature scheme which is existentially unforgeable against chosen message attack, and PKE = (CGen, Enc, Dec) be any IND-CCA2 secure encryption scheme supporting labels, and* $\mathsf{Com}(m, r)$ *be any statistically-hiding computationally-binding commitment scheme. Then the improved GMR scheme is a secure designated confirmer signature scheme, i.e., it satisfies the security requirements for verifiers, the signer, and the confirmer as specified in definitions 2, 3 and 4.*

6 A New DCS Scheme without Public Key Encryption

In this section, we propose a new generic DCS scheme, which is not only more efficient but also does not rely on any public key encryption. The basic idea is to exploit a *confirmer commitment* scheme, first introduced by Michels and Stadler in [30]. The difficulty, however, lies in realizing the invisibility in this setting, since Michels-Stadler DCS schemes were broken by Camenisch and Michels [10]. In our construction, we take a new approach to this problem by requiring the signer to issue a partial proof showing that a confirmer commitment is delegated to a specific signature and signer. To confirm or disavow an alleged signature, we extensively exploit the zero-knowledge protocols proposed by Korusawa and Heng [29] for their undeniable signatures.

Again, our scheme is just described for a short message digest $m \in [\rho]$, where ρ is a prime. To sign an arbitrary message $M \in \{0,1\}^*$ we can exploit a collision-free hash function $H_1 : \{0,1\}^* \rightarrow [\rho]$ and then use $H_1(M)$ to replace m in the following description.

- DCGen: The signer S generates a key pair $(sk_S, pk_S) \leftarrow \mathsf{SGen}(1^\lambda)$ for any secure digital signature scheme DSS=(SGen, Sig, Ver). The confirmer C chooses a group Γ of prime order ρ with a generator δ, and generates a key pair $(sk_C = x, pk_C = \gamma = \delta^x)$ by selecting a random number $x \in_R [\rho]$.
- Sign: To sign a message $m \in [\rho]$, the signer S first selects a random number $r \in [\rho]$, then computes $d_1 = \delta^r$, $d_2 = \gamma^{r+m}$, and $\sigma^* = \mathsf{Sig}(d_1||d_2||pk_S||pk_C, sk_S)$. After that, S with randomness r provides a non-interactive proof π_0 showing that $(\delta, d_1, \gamma, d_2\gamma^{-m})$ is a Diffie-Hellman (DH) tuple. That is,

$$\pi_0 = SPK\{(r,x) : (d_1 = \delta^r \wedge d_2\gamma^{-m} = \gamma^r) \vee (\gamma = \delta^x \wedge d_2\gamma^{-m} = d_1^x)\}(pk_S||pk_C).$$

  The basic signature is $\sigma = (\sigma^*, d_1, d_2, \pi_0)$.
- Verify: On input a basic signature $\sigma = (\sigma^*, d_1, d_2, \pi_0)$ and a message m, it outputs Accept if σ^* is the signer's valid signature on (d_1, d_2, pk_S, pk_C) and π_0 is a valid proof showing that $(\delta, \gamma, d_1, d_2\gamma^{-m})$ is a DH-tuple. Otherwise, it outputs $\perp$.
- Extract: On input an alleged DCS message-signature pair $(m, \sigma' = (\sigma^*, d_1, d_2, \pi_1))$ w.r.t. pk_S and pk_C, it outputs a non-interactive proof π_0 using the confirmer's private key x, if $(\delta, \gamma, d_1, d_2\gamma^{-m})$ is a DH-tuple. Otherwise, it outputs $\perp$.

– ConfirmedSign: To generate a DCS $\sigma' = (\sigma^*, d_1, d_2, \pi_1)$ for message $m \in [\rho]$, the signer first produces d_1, d_2 and σ^* as in the Sign procedure by selecting a random number r. Then, the signer S provides a non-interactive proof π_1 showing that he or she knows the discrete logarithm of d_1 to the base δ. That is,
$$\pi_1 = SPK\{r : d_1 = \delta^r\}(d_2||pk_S||pk_C).$$
We call $(m, \sigma' = (\sigma^*, d_1, d_2, \pi_1))$ is an *alleged* DCS message-signature pair w.r.t. pk_S and pk_C, if σ^* is a valid signature on $d_1||d_2||pk_S||pk_C$ w.r.t. public key pk_S, and π_1 is a valid signature proof of knowledge (SPK) [9] for $d_1 = \delta^r$ w.r.t. message $d_2||pk_S||pk_C$. Finally, S performs the interactive version of π_0 with a verifier V to show that $(\delta, d_1, \gamma, , d_2\gamma^{-m})$ is a DH-tuple, i.e.,
$$\pi_0' = PK\{(r, x) : (d_1 = \delta^r \wedge d_2\gamma^{-m} = \gamma^r) \vee (\gamma = \delta^x \wedge d_2\gamma^{-m} = d_1^x)\}.$$
– Confirm: For an alleged DCS message-signature pair $(m, \sigma' = (\sigma^*, d_1, d_2, \pi_1))$ with respect to pk_S and pk_C, the confirmer C checks if $(\delta, d_1, \gamma, , d_2\gamma^{-m})$ is a DH-tuple. If not, C performs Disavowal protocol. If yes, using its private key x the confirmer C runs the interactive protocol π_0' (see above) with a verifier V.
– Disavowal: To disavow an alleged DCS message-signature pair $(m, \sigma' = (\sigma^*, d_1, d_2, \pi_1))$ w.r.t. pk_S and pk_C, where $(\delta, d_1, \gamma, , d_2\gamma^{-m})$ is *not* a DH-tuple, using its private key x the confirmer C performs the following interactive protocol with a verifier V:
$$\pi_2' = PK\{(r, x) : (d_1 = \delta^r \wedge d_2\gamma^{-m} \neq \gamma^r) \vee (\gamma = \delta^x \wedge d_2\gamma^{-m} \neq d_1^x)\}.$$

Note that in the above specification, $d_2||pk_S||pk_C$ is particularly embedded in the partial proof π_1. The purpose is to prevent another signer from re-using (d_1, d_2, π_1). Otherwise, invisibility may be compromised. The implementation details and the security proof of the above DCS scheme can be found in the full version of this paper [37]. The following theorem summarizes the security result on this DCS scheme.

Theorem 2. *Let DSS = (SGen, Sig, Ver) be any signature scheme which is existentially unforgeable against chosen message attack, and $\Gamma = \langle\delta\rangle$ be a group in which the Decisional Diffie-Hellman (DDH) problem is intractable. Then the above DCS scheme without public key encryption is a secure designated confirmer signature scheme, i.e., it satisfies the security requirements for verifiers, the signer, and the confirmer as specified in definitions 2, 3 and 4.*

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digital Signatures. In: *Proc. of Advances in Cryptology - EUROCRYPT '98*, LNCS 1403, pp. 591-606. Springer-Verlag, 1998.
2. N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digital Signatures. *IEEE Journal on Selected Areas in Communications*, 18(4): 591-606, 2000.

3. G. Ateniese. Efficient Verifiable Encryption (and Fair Exchange) of Digital Signature. In: *Proc. of ACM Conference on Computer and Communications Security* (*CCS '99*), pp. 138-146. ACM Press, 1999.
4. F. Bao, R.H. Deng, and W. Mao. Efficient and Practical Fair Exchange Protocols with Off-line TTP. In: *Proc. of IEEE Symposium on Security and Privacy*, pp. 77-85, 1998.
5. M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In: *Proc. of the 1st ACM Conf. on Computer and Communications Security* (*CCS '93*), pp. 62-73. ACM press, 1993.
6. F. Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In: *Proc. of Advances in Cryptology - EUROCRYPT '00*, LNCS 1807, pp. 431-444. Springer-Verlag, 2000.
7. J. Boyar, D. Chaum, I. Damgard and T. Pedersen. Convertible Undeniable Signatures. In: *Proc. of Advances in Cryptology - CRYPTO'90*, LNCS 537, pp. 189-208, Springer-Verlag, 1990.
8. C. Boyd and E. Foo. Off-line Fair Payment Protocols Using Convertible Signatures. In: *Proc. of Advances in Cryptology - ASIACRYPT '98*, LNCS 1514, pp. 271-285. Springer-Verlag, 1998.
9. J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In: *Proc. of Advances in Cryptology - CRYPTO '97*, LNCS 1294, pp. 410-424. Springer-Verlag, 1997.
10. J. Camenisch and M. Michels. Confirmer Signature Schemes Secure against Adaptive Adversaries. In: *Proc. of Advances in Cryptology - EUROCRYPT '00*, LNCS 1870, pp. 243-258. Springer-Verlag, 2000.
11. J. Camenisch and V. Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: *Proc. of Advances in Cryptology - CRYPTO '03*, LNCS 2729, pp. 126-144. Springer-Verlag, 2003. Full version of this paper is available at `http://shoup.net/papers/`.
12. D. Chaum and H. van Antwerpen. Undeniable Signatures. In: *Proc. of Advances in Cryptology - CRYPTO'89*, LNCS 435, pp. 212-216, Springer-Verlag, 1989.
13. D. Chaum. Designated Confirmer Signatures. In: *Proc. of Advances in Cryptology - EUROCRYPT '94*, LNCS 950, pp. 86-91, Springer-Verlag, 1994.
14. L. Chen. Efficient Fair Exchange with Verifiable Confirmation of Signatures. In: *Proc. of Advances in Cryptology - ASIACRYPT '98*, LNCS 1514, pp. 286-299. Springer-Verlag, 1998.
15. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplied Design of Witness Hiding Protocols. In: *Proc. of Advances in Cryptology - CRYPTO '94*, LNCS 839, pp. 174-187. Springer-Verlag, 1994.
16. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: *Proc. of Advances in Cryptology - CRYPTO '98*, LNCS 1462, pp. 13-25. Springer-Verlag, 1998.
17. R. Cramer and V. Shoup. Signature Schemes based on the Strong RSA Assumption. In: *Proc. of the 6th ACM Conf. on Computer and Communications Security* (*CCS '99*), pp. 46-51. ACM press, 1999.
18. R. Cramer, I. Damgård, and P. MacKenzie. Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In: *Proc. of PKC '00*, LNCS 1751, pp. 354-373. Springer-Verlag, 2000.
19. I. Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: *Proc. of Advances in Cryptology - EUROCRYPT '00*, LNCS 1807, pp. 418-430, Springer-Verlag, 2000.

20. D. Dolev, D. Dwork, and N. Naor. Non-meallleable cryptography. In: *SIAM Journal on Computing*, 2000, 30(2): 391-437.
21. S. D. Galbraith and W. Mao. Invisibility and Anonymity of Undeniable and Confirmer Signatures. In: *Proc. of CT-RSA '03*, LNCS 2612, pp. 80-97. Springer-Verlag, 2003.
22. J. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free Optimistic Contract Signing. In: *Proc. of Advances in Cryptology - CRYPTO '99*, LNCS 1666, pp. 449-466. Sprnger-Verlage, 1999.
23. R. Gennaro, S. Halevi, and T. Rabin. Secure Hash-and-Sign Signatures without the Random Oracle. In: *Proc. of Advances in Cryptology - EUROCRYPT '99*, LNCS 1592, pp. 123-139. Springer-Verlag, 1999.
24. R. Gennaro. Multi-trapdoor Commitments and Their Applications to Proofs of Knowledge Secure Under Concurrent Man-in-the-Middle Attacks. In: *Advances in Cryptology - CRYPTO '04*, LNCS 3152, pp. 220-236. Springer-Verlag, 2004.
25. C. Gentry, D. Molnar, and Z. Ramzan. Efficient Designated Confirmer Signatures without Random Oracles or General Zero-knowledge Proofs. In: *Advances in Cryptology - ASIACRYPT 2005*, LNCS 3788, pp. 662-681. Springer-Verlag, 2005.
26. O. Goldreich and A. Kahan. How to Construct Constant-Round Zeroknowledge Proof Systems for NP. *Journal of Cryptology*, 9(3): 167-189, 1996.
27. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-message Attack. *SIAM Journal of Computing*, 17(2): 281-308, 1988.
28. S. Goldwasser and E. Waisbard. Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes. In: *Proc. of Theory of Cryptography TCC '04*, LNCS 2951, pp. 77-100, Springer-Verlag, 1996.
29. K. Kurosawa and S.-H. Heng. 3-Move Undeniable Signature Scheme. In: *Proc. of Advances in Cryptology - EUROCRYPT '05*, LNCS 3494, pp.181-197. Springer-Verlag, 2005.
30. M. Michels and M. Stadler. Generic Constructions for Secure and Efficient Confirmer Signature Schemes. In: *Proc. of Advances in Cryptology - EUROCRYPT '98*, LNCS 1403, pp. 406-421. Springer-Verlag, 1998.
31. M. Michels and M. Stadler. Efficient Convertible Undeniable Signature Schemes. In: *Proc. of 4th Annual Workshop on Selected Areas in Cryptography (SAC '97)*, pp. 231-244, 1997.
32. J. Monnerat and S. Vaudenay. Chaum's Designated Confirmer Signature Revisited. In: *Proc. of Information Security* (*ISC '05*), LNCS 3650, pp. 164-178. Springer-Verlag, 2005.
33. T. Okamoto. Designated Confirmer Signatures and Public Key Encryption Are Equivalent. In: *Proc. of Advances in Cryptology - CRYPTO '94*, LNCS 839, pp. 61-74. Springer-Verlag, 1994.
34. P. Paillier. Public Key Cryptosystems based on Composite Degree Residuosity Classes. *Proc. of Advances in Cryptology - EUROCRYPT '99*, LNCS 1592, pp. 223-238. Springer-Verlag, 1999.
35. T.P. Pedersen. Non-interactive and Information-theoretic Secure Verifiable Secret Sharing. In: *Proc. of Advances in Cryptology - CRYPTO '91*, LNCS 576, pp. 129-140. Springer-Verlag, 1992.
36. C.P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3): 161-174, 1991.
37. G. Wang, J. Baek, D.S. Wong, and F. Bao. On the Generic and Efficient Constructions of Secure Designated Confirmer Signatures. Full version of this paper is available from the authors or Cryptology ePrint Archive.