Invisible Designated Confirmer Signatures without Random Oracles

Victor K. Wei

Dept. of Information Engineering, The Chinese Univ. of Hong Kong, Hong Kong
kwwei@ie.cuhk.edu.hk

October 11, 2006

Abstract. We construct the first $O(1)$-size designated confirmer signatures (DCS) with security in the state-of-the-art model of Camenisch and Michels, Eurocrypt 2000, without random oracles. In particular, we achieve the security notion called the "invisibility of signature" therein.

1 Introduction

Chaum [9] introduced the DCS (Designated Confirmer Signature). The signature verification requires the interaction with a confirmer who was designated by the signer when the signature was created. The motivation was to split the power to sign and the power to confirm in order to mitigate the overpower of the signer. Several applications benefit from such a power splitting [9, 3].

T. Okamoto [20] gave a formal security model for DCS, and a polynomial equivalence reduction between DCS and public-key encryption. Camenisch and Michels [8] presented an upgraded DCS security model which included, among other things, the adaptive chosen confirmation attacker who can query a confirmation oracle about the validity of adaptively chosen DCS candidates. [8] also gave concrete instantiations, using the RSA signature and the Cramer-Shoup encryption. The confirmation and disavowal were not very efficient as they involved double discrete logarithms or range proof [7].

Goldwasser and Waisbard [16] and Gentry, et al. [15] presented DCS without random oracles. [15]'s DCS has $O(1)$-size. However, [16, 15] did not achieve the security notion of the *invisibility of signature* [8]. In a nutshell, the invisibility of the signature means that two DCS's corresponding to two messages are indistinguishable before confirmation. It essentially also means that distinguishing the validity/invalidity of a putative DCS is hard. This is a security notion motivated by zero-knowledge protocols.

[16, 15] argued such a requirement is unnecessarily strong and costly to attain.

We think it is still better to have such a requirement to defend against adaptive chosen confirmation attackers such as the well-known *signature transformation attackers* introduced in [8]. Furthermore, we show in this paper how to attain the invisibility efficiently without trading off other security notions. In signature transformation attacks the attackers transform the triple (putative DCS, message, signer public key) into another triple in such a way that their respective validity/invalidity are related. By querying the transformed triple to the confimration oracle, the attacker learns the validity of the transformed triple, and consequently learns that of the putative DCS. See [8] for the definition and several examples of signature transform attacks. In Section 5 we apply this attack on some DCS's in [16, 15]. Since [16, 15] did not claim the invisibility of the signature, our attacks are "beyond their model" attacks. Their DCS's remain secure in their own models.

Our **Contributions** are

1. We construct the first $O(1)$-size designated confirmer signatures (DCS) secure without random oracles in the strong security model of Camenisch and Michels [8], which includes the invisibility of signatures is achieved.
2. We apply [8]'s signature transformation attacks on [16, 15] to show they do not have the invisibility of signatures. However, [16, 15] did not claim the invisibility of signatures. Their DCS's remain secure in their own models.
3. We construct an $O(1)$-size undeniable signature secure without random oracles. Among the state-of-the-art security notions it attains is the invisibility of the signature. It is also interesting that this undeniable signature is cast in the bilinear subgroup (gap Diffie-Hellman group with non-trivial subgroups) which received invigorated resarch interests recently [6, 18, 2].

Related results. Camenisch and Michels '00 [8] presented a generic construction of DCS, which roughly proceeds as follows: The DCS is $\sigma = \mathsf{Enc}(\mathsf{pk}_C, \sigma' = \mathsf{Sign}(\mathsf{sk}_S, M))$, where Enc is a secure cipher, Sign a secure signature, sk_S is signer's private key, and pk_C is confirmer's public key. To confirm, the confirmer decrypts to obtain σ', and then conduct a (concurrent) zero-knowledge proof that $\sigma = \mathsf{Enc}(\mathsf{pk}_C, \sigma')$ and $\sigma' = \mathsf{Sign}(\mathsf{sk}_S, M)$. To disavow, the confirmer conduct (concurrent) zero-knowledge proof that

σ is not a valid ciphertext, or that it decrypts to σ' but σ' is not a valid signature. [8] gave the following theorem:

Theorem 1. *[8] If* Sign *is existentially unforgeable under an adaptive chosen-message (ACM) attack and* Enc *is chosen ciphertext attacker (CCA) secure, then the above construction constitutes a secure DCS.*

Our intuitions. Recently, Gentry, et al. [15] presented an efficient DCS without random oracles. Their generic construction uses, as building blocks, an arbitrary signature secure without random oracles, a suitable commitment, and an arbitrary public-key encryption secure without random oracles. When the Pedersen commitment and the Cramer-Shoup encryption is used with an arbitrary signature, the resulting DCS has $O(1)$-size and also has efficient coefficients. [15] did not achieve the invisibility of signatures. It is the main result of this paper to modify and upgrade [15]'s DCS to attain this security notion. The resulting DCS remains efficient, $O(1)$ in size with similarly efficient coefficients.

Our modification is a small one. [15] used the Cramer-Shoup encryption as a black box. We open the black box slightly to add more inputs to the hashing used inside. This modification is sufficient to ward off the signature transformation attacks and other adaptive chosen confirmation attacks on the invisibility. Details are in the remains of the paper.

2 Security model

We follow the security model of DCS in Camenisch and Michels [8]. Brief summaries below. Consult [8] for further details.

Syntax. A *designated confirmer signature (DCS)* is a tuple ($CKGS$, $CKGC$, $CSig$, $Eviden$=($CVerC$, $CVerV$), $CConv$, $COVer$) where

1. (Key Generation) $CKGS$ (resp. $CKGC$) accepts input the security parameter 1^{λ_s} to output signer key pair $(\mathsf{sk}_S, \mathsf{pk}_S)$ (resp. confirmer key pair $(\mathsf{sk}_C, \mathsf{pk}_C)$).
2. (Signing Protocol) $CSig$ accepts inputs message m, signer key pair $(\mathsf{sk}_S, \mathsf{pk}_S)$, confirmer public key pk_C, to output a signature σ.
3. (Evidentiation Protocol) $Eviden = (CVerC(\mathsf{sk}_C), CVerV)(m, \sigma, \mathsf{pk}_S, \mathsf{pk}_C)$ is a pair of interactive protocols corresponding to (Confirmer $CVerC$, Verifier $CVerV$) with common inputs m, σ, pk_S, pk_C, and private input to Confirmer is sk_C. At the conclusion of interactions, Verifier outputs 1/0, for $\mathsf{confirmed/disavowed}$.

4. (Selective Convertibility Protocol) Algorithm $CConv$ accepts inputs m, σ, pk_S, sk_C, pk_C, to output an ordinary signature s or NULL.
5. (Verification of Ordinary Signature Protocol) denoted $COVer$.

Correctness, validity, and security notions.

1. *Correctness of Evidentiation.* The Evidentiation Protocol is complete and correct.
2. *Validity of evidentiation.* With an honestly generated DCS in common inputs of the Evidentiation Protocol, no PPT protocol $CVerC^*$ can cause an honest $CVerV$ to output 0 with non-negligible probability; and with a string not corresponding to an honestly generated DCS is in the common inputs of the Evidentiation Protocol, no PPT protocol $CVerC^*$ can cause an honest $CVerV$ to output 1 with non-negligible probability.
3. *Security for the signer.* The DCS is *unforgeable* if no PPT attacker can deliver a valid DCS in the standard unforgeability game after making queries to the $CSig$ oracle. The attacker has sk_C. Naturally, the delivered DCS cannot be the output of a $CSig$ oracle query.
4. *Invisibility of signature (security for confirmers).* The DCS is *invisible* if no PPT adversary $\mathcal{A}$ can win the following game with probability non-negligibly over half: The simulator $\mathcal{B}$ sets up, gives all public keys and sk_S to $\mathcal{A}$. In arbitrary interleaf, $\mathcal{A}$ queries $CVerC$, $CConv$, generates signer public keys for these queries even not by invoking $CKGS$. At a certain point, $\mathcal{A}$ selects a message m_1. $\mathcal{B}$ selects a random message m_0, flips a fair coin b and sends the *gauntlet DCS*: $\sigma_{ga} = CSig(m_b, \mathsf{sk}_S, pk_S, \mathsf{pk}_C)$. At the end, $\mathcal{A}$ returns $\hat{b}$. $\mathcal{A}$ *wins* the game if $\hat{b} = b$ and σ_{ga} has never been queried to $CVerC$.
5. *Non-Transferability (of Evidentiation).* After the conclusion of Evidentiation, the verifier cannot convince a third party of the validity/invalidity of the signature.

Giving sk_S to the attacker in the invisibility definition models *forward security*: If the signer's long-term secret sk_S is stolen some time in the future, the validity/invalidity of the DCS should remain indistinguishable. Another, also standard, way to specify the invisibility experiment is to have $\mathcal{A}$ generates both m_0 and m_1.

Remark on non-transferability and un-impesonation: In [8], the non-transferability is defined in terms of the simulatability of the evidentiation transcript by the adversary without sk_C. It can usually be achieved by using concurrent zero-knowledge proofs in Protocol *Eviden*. We do not consider weakened notions such as *un-impersonation* from the literature which means $\mathcal{A}$ cannot convince a third party using Protocol *Eviden*. Note non-transferability essentially means $\mathcal{A}$ cannot convince a third party using *any* correct and sound interactive protocol.

Definition 1. *The DCS is* secure *if it has* correctness of evidentiation*,* correctness of conversion*,* validity of evidentiation*, and is* unforgeable*,* non-transferable*,* invisible*.*

We re-iterate Okamoto [20]'s equivalence reduction between DCS and pubic-key encryption below:

Theorem 2. (Okamoto [20]'s Theorem 3) *There exists an M-secure DCS if and only if there exists a secure public-key encryption.*

We briefly summarize [20]'s proof: Given a secure public key encryption encryption $\mathsf{Enc}(\mathsf{pk}, m)$, construct an M-secure signature $\mathsf{Sign}(\mathsf{sk}, m)$ using one-way functions. Then construct a DCS $\sigma = \mathsf{Enc}(\mathsf{pk}_C, \mathsf{Sign}(\mathsf{sk}_S, m))$. Note the security of the encryption, i.e. the indistinguishability of the encryptions of two plaintexts, implies the invisibility of the DCS. Given an M-secure DCS, encrypt the one-bit plaintext $b \in \{0, 1\}$ to the public key pk_C as $CSig(\mathsf{sk}_S,\ \mathsf{pk}_C,\ \mathsf{const}_b)$. The invisibility of the DCS implies indistinguishability of the ciphertexts.

3 Preliminaries

3.1 Intractability assumptions

The *strong RSA problem* is, given $n = pq$, p and q are unknown primes, $z \in \mathbb{Z}_n$, compute (A, e), $e \geq 3$, satisfying $A^e = z \pmod{n}$. The *strong RSA assumption* is that no PPT algorithm can solve a random instance of the strong RSA problem with non-negligible probability.

The *decision composite residuosity (DCR) assumption* [21] is that given n it is hard to distinguish random elements of $\mathbb{Z}^*_{n^2}$ from random elements of all n-th powers of elements in $\mathbb{Z}^*_{n^2}$.

The *decisional Diffie-Hellman (DDH) problem* is, given g, g^a, g^b, g^c, distinguish $c = ab$ from random. The *DDH assumption* is that no PPT algorithm can solve a random instance of the DDH problem with probability non-negligibly over half.

3.2 Survey

We briefly summarize relevant literatures. Consult original references for details.

Cramer-Shoup encryption [11]:

1. *Setup*: $\mathsf{sk} = (x, x_2, y_1, y_2, z)$, $\mathsf{pk} = (d_1 = g_1^{x_1} g_2^{x_2},\ d_2 = g_1^{y_1} g_2^{y_2},\ d_3 = g_1^z)$.
2. *Encrypt*: $\mathsf{ctxt} = (u_1 = g_1^r,\ u_2 = g_2^r,\ u_3 = \mathsf{ptxt} \cdot d_3^r,\ u_4 = (d_1 d_2^\alpha)^r)$ where r is randomly selected and $\alpha = Hash(u_1,\ u_2,\ u_3)$.
3. *Decrypt*: Verify $u_4 = g_1^{x_1+\alpha y_1} g_2^{x_2+\alpha y_2}$ before outputting $\mathsf{ptxt} = u_3 u_1^{-1}$.
4. *Security*: The encryption is secure against adaptive chosen ciphertext attackers (without random oracles) provided the decisional Diffie-Hellman (DDH) assumption holds.

Paillier system and partial discrete logarithm: Given $n = pq$, p and q are all primes, and $g_0 = n+1$, it is computationally easy to compute the *partial discrete logarithm (PDL)*: That is, given $m \in \mathbb{Z}_{n^2}$, compute x and y satisfying $g_0^x y^n = m$. Here is the PDL algorithm: Compute $m' = m^{(p-1)(q-1)} = g_0^{x(p-1)(q-1)} y^{n(p-1)(q-1)} = (n+1)^{x(p-1)(q-1)} = x(p-1)(q-1)n + 1 \in \mathbb{Z}_{n^2}$. Then compute $x = ((m'-1)/n)((p-1)(q-1))^{-1} \in \mathbb{Z}_n$.

Four-move concurrent zero-knowledge protocols: We very briefly summarize this advanced topic. Consult original papers [14, 22] for details. In a nutshell, concurrent zero-knowledge protocols are zero-knowledge protocols that can be concurrently compositioned while retaining zero-knowledge. This property requires that the protocol transcript (without timeline) can be simulated by the Verifier in an indistinguishing way.

While the general topic of concurrent zero-knowledge protocols, especially three-move ones, remain highly advanced, there is a well-known method [22] to convert a typical three-move zero knowledge protocol into a four-move concurrent zero-knowledge protocol as follows: Denote a typical three-move zero-knowledge protocol as $(D,\ c,\ z)$ for (commit, challenge, response). The four moves of the converted concurrent zero-knowledge protocol consists of $(c' = H(c),\ D,\ c,\ z)$. In the first move, Verifier selects challenge c, sends its hashed value c'. In the second move, Prover sends the commitment D. In the third move, Verifier sends c. In the fourth move, Prover checks $c' = H(c)$ before sending the response z. Finally, Verifier checks before outputting 0 or 1. The protocol transcript (when there is no timeline ordering of the four moves) can be simulated by the Verifier as follows: Compute Move-3, Move-4, Move-2, Move-1, in that order.

3.3 Gentry, et al. [15]'s DCS

The main DCS in [15] on message m is $\sigma' = (\sigma^*, \phi, c)$, where

$$\begin{aligned} \phi &= Commit(m, R) \\ c &= \mathsf{Enc}(\mathsf{pk}_C, R) \\ \sigma^* &= \mathsf{Sign}(\mathsf{sk}_S, (\phi, c, \mathsf{pk}_S)) \end{aligned} \tag{1}$$

Theorem 3. *[15] The above DCS is* secure *provided the signature scheme* Sign *is existentially unforgeable against adaptive chosen message attackers, the commitment scheme* $Commit$ *is secure (computationally binding and statistically hiding with zero-knowledge proof of knowledge for committed value secure against cheating verifier), and the public-key encryption scheme* Enc *is IND-CCA2 secure.*

[15] presented a particularly efficient instantiation where Pedersen's commitment, the Cramer-Shoup encryption, and an arbitrary secure signature is used:

$$\phi = g^m h^R \in QR_{n^2}$$
$$c = (u_1, u_2, u_3, u_4) = (g_1^r, g_2^r, d_3^r g_0^R, (d_1 d_2^\alpha)^r) \in QR_{n^2}^4$$

where $\alpha = Hash(u_1,\ u_2,\ u_3)$, order$(h) = n$, $g_0 = n + 1$. The confirmer public key pk_C consists of $d_1 = g_1^{x_1} g_2^{x_2}$, $d_2 = g_1^{y_1} g_2^{y_2}$, $d_3 = g_1^z$. Its private key is $\mathsf{sk}_C = (x_1,\ x_2,\ y_1,\ y_2,\ z)$. Note the confirmer can compute the partial discrete logarithm with base g_0, and therefore can decrypt R.

4 Constructing DCS

We modify Gentry, et al. [15]'s DCS, reviewed in Section 3.3, to upgrade it with invisibility. The modification is simple: merely add more parameters to the input of the public-key encryption.

Invisible DCS: $\mathsf{DCS}_{\mathrm{GMR+}}$:

Key Generation: $CKGS(1^{\lambda_s}) \mapsto (\mathsf{sk}_S, \mathsf{pk}_S)$ and $CKGC(1^{\lambda_s}) \mapsto (\mathsf{sk}_C, \mathsf{pk}_C)$. $(\mathsf{sk}_S, \mathsf{pk}_S)$ corresponds to an arbitrary signature scheme which is existentially unforgeable against adaptive chose message attackers without random oracles. For example, the Cramer-Shoup signature [12], Boneh, et al. [5], Wei, et al. [23].

$$\mathsf{sk}_C = (x_1, x_2, y_1, y_2, z_1, z_2),$$
$$\mathsf{pk}_C = (d_1 = g_1^{x_1} g_2^{x_2}, d_2 = g_1^{y_1} g_2^{y_2}, d_3 = g_1^{z_1} g_2^{z_2}, n, g_1, g_2, g, h, g_0)$$

where g_1, g_2, g, h, $g_0 \in QR_{n^2}$ are fairly generated, order(g_1) = order(g_2) = order$(g) = np'q'$, order(h) = order$(g_0) = n$, $n = pq$, $(2p'+1)(2q'+1)$ with p, q, p', q' being primes. For example, $g_0 = n + 1$.

Signing: $CSig(m, \mathsf{sk}_S, \mathsf{pk}_S, \mathsf{pk}_C) \mapsto \sigma$. Select random $R \in \mathbb{Z}_n$, $\bar{r} \in [1, n^2/4]$, output $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ where

$$\sigma_1 = g^m h^R,\ \sigma_2 = \mathsf{Enc}(\mathsf{pk}_C, R) = (u_1, u_2, u_3, u_4), \tag{2}$$

$$\sigma_3 = \mathsf{Sign}(\mathsf{pk}_S, (\sigma_1, \sigma_2)),\ u_1 = g_1^{\bar{r}}, \tag{3}$$

$$u_2 = g_2^{\bar{r}},\ u_3 = d_3^{\bar{r}} g_0^{\bar{R}},\ v = (d_1 d_2^{\alpha})^{\bar{r}} \text{ where} \tag{4}$$

$$\alpha = \bar{H}(u_1, u_2, u_3, \mathsf{pk}_S, \mathsf{pk}_C, \sigma_1, m) \tag{5}$$

Confirmation and disavowal: $(CVerC(\mathsf{sk}_C), CVerV)(m, \sigma, \mathsf{pk}_S, \mathsf{pk}_C, \mathsf{param})$: Note $\sigma = (\sigma_1, \sigma_2 = (u_1, u_2, u_3, v), \sigma_3)$. $CverC$ checks σ_3 is a valid signature by signer on (σ_1, σ_2), checks $v = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$, then computes R equals the partial discrete logarithm of $u_3 u_1^{-z_1} u_2^{-z_2}$ with base g_0, checks $u_3 u_1^{-z_1} u_2^{-z_2} = g_0^R$ and $g^m h^R = \phi$. If all pass, send signal to confirm. Else, send signal to disavow.

To confirm, $CVerC$ proves to $CVerC$ of the following in concurrent zero-knowledge:

$$\begin{aligned} &ConZKP\{(x_1, x_2, y_1, y_2, z_1, z_2, R) : v = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} \\ &\wedge\ u_3 = u_1^{z_1} u_2^{z_2} g_0^R\ \wedge\ d_1 = g_1^{x_1} g_2^{x_2}\ \wedge\ d_2 = g_1^{y_1} g_2^{y_2}\ \wedge\ d_3 = g_1^{z_1} g_2^{z_2}\} \end{aligned} \tag{6}$$

where α is computed according to Eq. (5). We instantiate (6) using a typical four-move concurrent zero-knowledge protocol in Appendix A.

To disavow: The DCS is automatically disavowed if σ_3 is not a signature by sk_S on (σ_1, σ_2). Else conduct the following concurrent zero-knowledge proof:

$$\begin{aligned} &ConZKP\{(x_1, x_2, y_1, y_2, z_1, z_2, R, p', q') : \\ &\quad d_1 = g_1^{x_1} g_2^{x_2}\ \wedge\ d_2 = g_1^{y_1} g_2^{y_2}\ \wedge\ d_3 = g_1^{z_1} g_2^{z_2} \\ &\quad \wedge\ (2p'+1)(2q'+1) = n\ \wedge\ (u_3^{-1} u_1^{z_1} u_2^{z_2} g_0^R)^{p'q'} = 1 \\ &\quad \wedge\ [v \neq u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}\ \vee\ g^m h^R \neq \phi\ \vee\ u_3^{-1} u_1^{z_1} u_2^{z_2} g_0^R \neq 1] \end{aligned} \tag{7}$$

A detailed instantiation of (7) is given in Appendix A.

This ends the specification of Protocol $\mathsf{DCS}_{\mathrm{SDH}}$. Below is the security reduction theorem, whose proof is sketched in Appendix B.

Theorem 4. *The* designated confirmer signature $\mathsf{DCS}_{\mathrm{GMR+}}$ *is* secure *provided the signature scheme* Sign *is existentially unforgeable against adaptive chosen message attackers, the DDH (decision Diffie-Hellman) assumption holds in* QR_{n^2}, *and* $\bar{H}$ *is a collision-resistant hash function.*

The theorem implies that $\mathsf{DCS}_{\mathrm{GMR}+}$ is secure without random oracles provided the latter two conditions also hold without random oracles. By choosing a signature secure without random oracles, $\mathsf{DCS}_{\mathrm{GMR}+}$ is secure without random oracles.

5 Signature transformation attacks

We apply [8]'s signature transformation attacks on [15, 16]'s DCS's. The attack consequence is to cryptanalyze the invisibility of the signatures. However, [15, 16] did not claim this security notion, our attacks in this Section are "beyond their model" attacks, meant to establish their DCS's indeed do not have the invisibility in our model. Their results remain secure in their own model.

5.1 Signature transformation attack on Gentry, et al. [15]

The DCS scheme was reviewed in Section 3.3 and Eq. (1). The attacker needs the following *attack hypotheses*:

1. Knowing the private key $\mathsf{sk}_{S'}$ of a signer $S' \neq S$.
2. Query access to a confirmation oracle which, upon common inputs including a message $\bar{m}$, a signer public key pk_S, and a putative DCS $\bar{\sigma}'$, will confirm or disavow the DCS $\bar{\sigma}'$. Except when the queried tuple $(\bar{m},\ \mathsf{pk}_{S'},\ \bar{\sigma}')$, $\bar{\sigma}' = (\bar{\sigma}^*,\ \bar{\phi},\ \bar{c})$, shares the same m, or the same pk_S, or the same σ^*, or the same ϕ as the attacker's target tuple. Note queries with the same $\bar{c} = c$ are allowed.

Attack consequence and procedure: Given a (message, signer private key, putative DCS) tuple, denoted $(m,\ \mathsf{pk}_S,\ \sigma')$, our attacker computes the validity of the putative DCS (i.e. distinguishes a valid DCS from a non-valid simulation DCS) and consequently cracks the security of the DCS scheme by cracking its transcript simulatability [15]. It does so by interacting once with the confirmation oracle with the following transformed tuple: $(\bar{m},\ \mathsf{pk}_{S'},\ \bar{\sigma}')$, $\bar{\sigma}' = (\bar{\sigma}^*,\ \bar{\phi},\ \bar{c})$, where $\bar{m} = m + 1$, $\mathsf{pk}_{S'}$ is from the attack hypotheses, and $\bar{c} = c$, $\bar{\phi} = \phi g$ (which corresponds to $\bar{r} = r$), and $\bar{\sigma}^* = \mathsf{Sign}(\mathsf{sk}_{S'},\ (\bar{\phi},\ \bar{c},\ \mathsf{pk}_{S'}))$. The transformed DCS has the same validity/invalidity as the pre-transformation DCS. Interacting with the confirmation oracle yields the validity/invalidity of the transformed DCS, and consequently the validity/invalidity of the original pre-transformation DCS.

Attack generalization. Replacing Equation (1) by $\phi = g^{H(m,\mathsf{pk}_S,\mathsf{pk}_C,c)}h^r$ is not a sufficient defense as we can achieve the same attack using

$$\bar{\phi} = \phi g^{H(\bar{m},\mathsf{pk}_S,\mathsf{pk}_C,c)-H(m,\mathsf{pk}_S,\mathsf{pk}_C,c)}$$

Other DCS schemes that use public-key encryption as a black-box building block, such as those in [16, 15] and elsewhere, may also risk signature transformation attacks. In fact, we demonstrate a signature transformation attack on [16] subsequently.

Attack mitigation: Follow Section 4 by adding more input parameters to the hash α such as in Eq. (5). Another mitigation is to encrypt more parameters, e.g. by changing c in Eq. (5) to $c = \mathsf{Enc}(\mathsf{pk}_C, (R, \phi, m, \mathsf{pk}_S, \mathsf{pk}_C)$. This is a more bandwidth-expensive remedy than that in Section 4). The tradeoff is that its reductionist security proof is somewhat easier.

Our results suggest that other schemes that use public-key encryption as a black-box building block, such as those in [16, 15] and elsewhere, should also use our easy mitigation technique: Open the black box slightly and add more parameters to the hash input or to the input of other kinds of *tag* generating mechanisms [1].

5.2 Signature transformation attack on Goldwasser, et al. [16]

. *Review.* We focus on the first concrete DCS in [16] which is based on the Cramer-Shoup signature [12] and the Cramer-Shoup encryption [11]. The Cramer-Shoup signature on message m is $\sigma' = (e, y', y)$,

$$y^e = xh^{H(x')}$$
$$x' = (y')^{e'}h^{-H(m)}$$

where the signer's public key is $\mathsf{pk}_S = (n,\ h,\ x,\ e')$, n is a product of two primes, e' and e' are distinct primes, h and x are random. The Goldwasser, et al.'s DCS is $\sigma = (\sigma_1 = e,\ \sigma_2 = y',\ \sigma_3 = \mathsf{Enc}(\mathsf{pk}_C,\ y)\ )$.

Attack Hypotheses. The attacker needs the following hypotheses:

1. Knowing the private key sk_S of the signer.
2. Query access to a confirmation oracle which, upon common inputs including a message $\bar{m}$, a signer public key $\mathsf{pk}_{S'}$, and a putative DCS $\bar{\sigma}$, will confirm or disavow the DCS $\bar{\sigma}$. Except when the queried tuple $(\bar{m},\ \mathsf{pk}_{S'},\ \bar{\sigma})$, $\bar{\sigma} = (\bar{e},\ \bar{y}',\ \bar{\sigma_3})$, shares the same m, or the same pk_S, or the same σ_1, or the same σ_2 as the attacker's target tuple. Queries with the same σ_3 are allowed.

3. The signature verification protocol does not check e is a prime. It only checks that it is within a certain range. This is a common practice in using the Cramer-Shoup encryption, e.g. [12, 16], to keep computational complexities low.

Attack consequence and procedure. The attacker can compute the validity/invalidity of a given putative DCS by interacting once with the confirmation oracle with the following transformed putative DCS: $\bar{\sigma} = (\bar{e}, \bar{y}', \sigma_3)$ on a new arbitrary message $\bar{m}$ for a new signer public key $\bar{\mathsf{pk}}_S = (n, \bar{h}, \bar{x}, e')$ where $\bar{x} = y^e$, $\bar{h} = y$, $\bar{x}' = (y')^{e'}\bar{h}^{-H(\bar{m})}$, $\bar{e} = e + H(\bar{x}')$. It is mechanical to verify that the transformed DCS has the same validity/invalidity as the pre-transformation DCS. Interacting with the confirmation oracle yields the validity/invalidity of the transformed DCS, and consequently the validity/invalidity of the original pre-transformation DCS.

Therefore, an adversary $\mathcal{A}$ can distinguish a valid signature from an invalid one by interacting with the confirmation oracle. However, [16] does not claim the indistinguishability between valid and invalid signatures, called the *invisibility of the signature* in [20, 8]. Our attack is beyond their security model. Their DCS remains secure in their own model.

Mitigation. Nevertheless, we suggest to include more parameters in the has inputs wherever possible to defend against signature transformation and potentially other attacks. For example, letting $x' = (y')^{e'}h^{-H(m,\mathsf{pk}_S,e,y')}$ or having even more parameters included in the hash inputs can contribute to enhanced security.

6 Invisible undeniable signature from bilinear subgroups

Generic undeniable signatures from DCS

Undeniable signatures [10] are DCS's where signer and confirmer are the same entity. Using techniques developed above, we can modify Laguillaumie, et al. [19]'s undeniable signature without random oracles to upgraded security model with signature invisibility and defense against signature transformation attackers. Consult original references for details of the security model.

1. *Setup.* The signer public key $\mathsf{pk} = (n, y_1, y_2, {}_1, d_2, d_3)$, $\mathsf{sk} = (x_1, x_2, \bar{x}_1, \bar{x}_2, \bar{y}_1, \bar{y}_2, z)$, where $y_1 = g^{x_1}$, $y_2 = g^{x_2}$, $d_1 = g_1^{\bar{x}_1} g_2^{\bar{x}_2}$, $d_2 = g_1^{\bar{y}_1} g_2^{\bar{y}_2}$, $d_3 = g_1^z$, n is a product of two safe primes p and q, pairings $\hat{\mathbf{e}} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, order$(\mathbb{G}_1) = n$, $g \in \mathbb{G}_1$, $g_1, g_2 \in \mathbb{Z}_{n^2}$, $g_0 = n + 1$.
2. *Sign.* Select random $R \in \mathbb{Z}_n$, compute $\sigma = (\sigma_1, \sigma_2)$, where $\sigma_1 = g^{1/(x_1+R+mx_2)}$, $\sigma_2 = \mathsf{Enc}(R) = (u_1, \cdots, u_4)$ with $u_1 = g_1^r$, $u_2 = g_2^r$, $u_3 = d_3^r g_0^R$, $u_4 = (d_1 d_2^\alpha)^r$, where $\alpha = Hash(u_1, u_2, u_3, m, \mathsf{pk}, \sigma_1)$.

3. *Confirm/disavow.* To confirm, prove the following concurrent zero-knowledge protocols:

$$CZK\{R : \hat{\mathbf{e}}(\sigma_1, y_1 y_2^m)\hat{\mathbf{e}}(\sigma_1, g)^R = \hat{\mathbf{e}}(g, g) \ \wedge \ u_3 = u_1^z g_0^R \ \wedge \ d_3 = g_1^z\}$$

To disavow, prove the following concurrent zero-knowledge protocol

$$\begin{aligned} &CZK\{(\bar{x}_1, \bar{x}_2, \bar{y}_1, \bar{y}_2, z, R') : d_1 = g_1^{\bar{x}_1} g_2^{\bar{x}_2} \\ &\quad \wedge \ d_2 = g_1^{\bar{y}_1} g_2^{\bar{y}_2} \ \wedge \ d_3 = g_1^z \ \wedge \ u_3 = u_1^z g_0^{R'} \\ &\quad \wedge \ [u_4 \neq u_1^{\bar{x}_1 + \alpha\bar{y}_1} u_2^{\bar{x}_2 + \alpha\bar{y}_2} \ \vee \ \hat{\mathbf{e}}(\sigma_1, y_1 y_2^m)\hat{\mathbf{e}}(\sigma_1, g)^{R'} \neq \hat{\mathbf{e}}(g, g)\}\end{aligned}$$

Note order$(g_0) = n$ in $\mathbb{Z}_{n^2}$. There is no need to prove for the *proof of range* that R (and R') lie in the interval $[0, n)$. Th invisibility of signature mainly follows the use of concurrent zero-knowledge protocols. The unforgeability of the undeniable signature can be proved similarly to [19]. Methods to instantiate a pairings group (or *gap Diffie-Hellman group*) $\mathbb{G}_1$ with a composite order n were described in Boneh, et al. [6] and Groth, et al. [17].

Generalization. The undeniable signature above combines Boneh, et al. [5]'s signature without random oracles and the famous Cramer-Shoup encryption [11] without random oracles. It can be modified into a DCS by separating the signing key (given to the signer) and the encryption key (given to the confirmer). But then the confirmer key, $\mathsf{pk}_C = (d_1, d_2, d_3)$, is dependent of the signer public key n, as the three entries lie in $\mathbb{Z}_{n^2}$. Although security is not compromised because the security of the Cramer-Shoup encryption reduces to the decisional Diffie-Hellman assumption in $\mathbb{Z}_{n^2}$ which continues to hold, this dependence is not desirable. If entries of pk_C are in $\mathbb{Z}_{\bar{n}^2}$ with $\bar{n} \neq n$, then inefficient range proofs may have to be used in the confirmation/disavowal protocol.

7 Conclusion

We have presented new constructions of DCS with invisibility. It remains to construct efficient and secure DCS where the confirmer is identity-based. **Acknowledgements**to Hong Kong Earmarked Grants 4232-03E and 4328-02E for sponsorship.

References

1. Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In *EUROCRYPT 2005*, pages 128–146, 2005.

2. Ben Adida and Douglas Wikström. How to shuffle in public. Cryptology ePrint Archive, Report 2005/394, 2005. http://eprint.iacr.org/.
3. N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. In *EUROCRYPT 1998*, pages 591–606, 1998.
4. Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO 2002*, pages 162–177, 2002.
5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proc. CRYPTO 2004*, pages 41–55. Springer-Verlag, 2004. Lecture Notes in Computer Science No. 3152.
6. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC 2005*, pages 325–341, 2005.
7. F. Boudot. Efficient proofs that a committed number lies in an interval. In *Eurocrypt'00*, pages 431–444, 2000.
8. J. Camenisch and M. Michels. Confirmer signature schemes secure against adaptive adversaries. In *Eurocrypt 2000*, pages 243–258. Springer-Verlag, 2000. LNCS No. 2729.
9. D. Chaum. Designated confirmer signatures. In *Eurocrypt'94*, pages 86–91. Springer-Verlag, 1994. LNCS No. 435.
10. D. Chaum and H. van Antwerpen. Undeniable signatures. In *Crypto'89*, pages 286–299, 1989.
11. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer-Verlag, 2002.
12. Ronald Cramer and Victor Shoup. Signature schemes based on the strong rsa assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.
13. Ivan Damgard, Kasper Dupont, and Michael Ostergaard Pedersen. Unclonable group identification. In *EUROCRYPT 2006*, pages 555–572, 2006.
14. C. Dwork, M. Naor, , and A. Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004. Also in STOC'98, p.409-418.
15. Craig Gentry, David Molnar, and Zulfikar Ramzan. Efficient designated confirmer signatures without random oracles or general zero-knowledge proofs. In *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 662–681. Springer-Verlag, 2005.
16. Shafi Goldwasser and Erez Waisbard. Transformation of digital signature schemes into designated confirmer signature schemes. In *TCC 2004*, volume 2951 of *LNCS*, pages 77–100. Springer-Verlag, 2004.
17. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. Cryptology ePrint Archive, Report 2005/290.
18. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT 2006*, pages 339–358, 2006.
19. Fabien Laguillaumie and Damien Vergnaud. Short undeniable signatures without random oracles: The missing link. In *INDOCRYPT 2005*, volume 3797 of *LNCS*, pages 283–296. Springer-Verlag, 2005.
20. T. Okamoto. Designated confirmer signatures and public-key encryption are equivalen. In *Proc. CRYPTO '94*, pages 61–74, 1994.
21. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT '99*, pages 223–238. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1592.
22. Mario Di Raimondo and Rosario Gennaro. New approaches for deniable authentication. In *ACM Conference on Computer and Communications Security*, pages 112–121, 2005. Also Eprint 2003/056.

23. Victor K. Wei and T. H. Yuen. More short signatures without random oracles, 2007. ePrint 2005/463.

A Elaborating ($CVerC$, $CVerV$) of $\mathsf{DCS}_{\mathrm{GMR+}}$

The confirmation (6) is composed of only multi-exponentiations. Instantiating it using a standard four-move concurrent zero-knowledge protocol is straightforward: (The common inputs are pk_S, pk_C, $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4 = (u_1, u_2, u_3, v))$. The private inputs of $CVerC$ are x_1, x_2, y_1, y_2, z_1, z_2, R) where R is computed from the partial discrete logarithm as in $CVerC$.

1. $CVerV$ selects random c', compute $c'' = Hash(c')$, sends c''.
2. $CVerC$ selects random $r_{x,1}$, $r_{x,2}$, $r_{y,1}$, $r_{y,2}$, $r_{z,1}$, $r_{z,2}$, compute and sends the following *commitments*:
$$D_v = u_1^{r_{x,1}+r_{y,1}\alpha} u_2^{r_{x,2}+r_{y,2}\alpha},\ D_u = u_1^{r_{z,1}} u_2^{r_{z,2}} g_0^{r_R},$$
$$D_1 = g_1^{r_{x,1}} g_2^{r_{x,2}},\ D_2 = g_1^{r_{y,1}} g_2^{r_{y,2}},\ D_3 = g_1^{r_{z,1}} g_2^{r_{z,2}},$$
3. $CVerV$ sends c'.
4. $CVerC$ checks $c'' = Hash(c')$, compute and sends the *responses*:
$$z_{x,1} = r_{x,1} - c'x_1,\ z_{x,2} = r_{x,2} - c'x_2,\ z_{y,1} = r_{y,1} - c'y_1,$$
$$z_{y,2} = r_{y,2} - c'y_2,\ z_{z,1} = r_{z,1} - c'z_1,\ z_{z,2} = r_{z,2} - c'z_2,\ z_R = r_R - c'R$$
5. $CVerV$ verifies the following before outputting 1: (If not all verified, output 0)
$$D_v = u_1^{z_{x,1}+z_{y,1}\alpha} u_2^{z_{x,2}+z_{y,2}\alpha} v^{c'},\ D_u = u_1^{z_{z,1}} u_2^{z_{z,2}} g_0^{z_R} u_3^{c'},$$
$$D_1 = g_1^{z_{x,1}} g_2^{z_{x,2}} d_1^{c'},\ D_2 = g_1^{z_{y,1}} g_2^{z_{y,2}} d_2^{c'},\ D_3 = g_1^{z_{z,1}} g_2^{z_{z,2}} d_3^{c'},$$

To instantiate the disavowal (7), $CVerC$ sends $\beta \in \{1,2,3\}$ indicating its wish to prove (Case 1) $v \neq u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$; (Case 2) $g^m h^R \neq \phi$; or (Case 3) $u_3^{-1} u_1^{z_1} u_2^{z_2} g_0^R \neq 1$. Further elaborations: Case 1)

$$\begin{aligned}
&ConZKP\{(x_1, x_2, y_1, y_2) : d_1 = g_1^{x_1} g_2^{x_2} \\
&\quad \wedge\ d_2 = g_1^{y_1} g_2^{y_2}\ \wedge\ v \neq u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}\} \\
&\Leftrightarrow ConZKP\{(x_1, x_2, y_1, y_2, s_0, s_1, s_2, s_3, s_4) : d_1 = g_1^{x_1} g_2^{x_2} \\
&\quad \wedge\ d_2 = g_1^{y_1} g_2^{y_2}\ \wedge\ T_0 = v^{-s_0} u_1^{s_1+s_2\alpha} u_2^{s_3+s_4\alpha} \\
&\quad \wedge\ T_1 = g_1^{x_1}\ \wedge\ T_2 = g_1^{y_1}\ \wedge\ T_3 = g_1^{x_2}\ \wedge\ T_4 = g_1^{y_2} \\
&\quad \wedge\ 1 = T_1^{x_1} g_1^{-s_1}\ \wedge\ 1 = T_2^{y_1} g_1^{-s_2}\ \wedge\ 1 = T_3^{x_2} g_1^{-s_3}\ \wedge\ 1 = T_4^{y_2} g_1^{-s_4}\}
\end{aligned}$$

The four-move concurrent zero-knowledge protocol is

1. $CVerV$ sends $c'' = Hash(c')$.
2. $CVerC$ sends $T_0, \cdots, T_4$, and

$$\begin{array}{l} D_1' = g_1^{r_{x,1}} g_2^{r_{x,2}},\ D_2' = g_1^{r_{y,1}} g_2^{r_{y,2}},\ D_0 = v^{-r_0} u_1^{r_1+r_2\alpha} u_2^{r_3+r_4\alpha}, \\ D_1 = g_1^{r_{x,1}},\ D_2 = g_2^{r_{y,1}},\ D_3 = g_3^{r_{x,2}},\ D_4 = g_4^{r_{y,2}}, \\ D_5 = T_1^{r_{x,1}} g_1^{r_1},\ D_6 = T_2^{r_{y,1}} g_1^{r_2},\ D_7 = T_3^{r_{x,2}} g_1^{r_3},\ D_8 = T_4^{r_{x,2}} g_1^{r_4} \end{array}$$

   where $s_1 = s_0 x_1$, $s_2 = s_0 y_1$, $s_3 = s_0 x_2$, $s_4 = s_0 y_2$.
3. $CVerV$ checks $T_0 \neq 1$, sends c'.

The remaining moves are straightforward and omitted. Cases 2):

$$\begin{array}{l} ConZKP\{(z_1, z_2, R, p', q') : d_3 = g_1^{z_1} g_2^{z_2} \ \wedge\ (u_3^{-1} u_1^{z_1} u_2^{z_2} g_0^{R})^{p'q'} = 1 \\ \quad \wedge\ (2p'+1)(2q'+1) = n \ \wedge\ g^m h^R \neq \phi\} \\ \Leftrightarrow ConZKP\{(z_1, z_2, R, p', q') : d_3 = g_1^{z_1} g_2^{z_2} \ \wedge\ T_1 = g_1^{s_0} \\ \quad \wedge\ T_2 = u_1^{z_1} u_2^{z_2} g_0^{R} g_2^{s_2} \ \wedge\ T_3 = (u_3^{-1} T_2)^{p'} g^{-s_2} g_3^{s_0} \ \wedge\ 1 = T_1^{R} g_1^{-s_1} \\ \quad \wedge\ 1 = T_3^{q'} g_3^{-s_3} \ \wedge\ 1 = T_1^{p'} g_1^{-s_2} \ \wedge\ 1 = T_1^{q'} g_1^{-s_3} \ \wedge\ T_4 = T_1^{p'} g_4^{s_0} \\ \quad \wedge\ T_1^{n} = T_4^{4q'+2} T_1^{2q'+1} g_4^{-4s_3-2s_0} \ \wedge\ T_5 = (g^m \phi^{-1})^{s_0} h^{s_1}\} \end{array}$$

where $CVerC$ uses $s_1 = Rs_0$, $s_2 = s_0 p'$, $s_3 = s_0 q'$, and $CVerV$ checks $T_5 \neq 1$. The four-move concurrent zero-knowledge protocol is straightforward and omitted. The case 3):

$$\begin{array}{l} ConZKP\{(z_1, z_2, R, p', q') : d_3 = g_1^{z_1} g_2^{z_2} \ \wedge\ (u_3^{-1} u_1^{z_1} u_2^{z_2} g_0^{R})^{p'q'} = 1 \\ \quad \wedge\ (2p'+1)(2q'+1) = n \ \wedge\ u_3^{-1} u_1^{z_1} u_2^{z_2} g_0^{R} \neq 1\} \\ \Leftrightarrow ConZKP\{(z, R, p', q') : d_3 = g_1^{z_1} g_2^{z_2} \ \wedge\ T_1 = g_1^{s_0} \\ \quad \wedge\ T_2 = u_1^{z_1} u_2^{z_2} g_0^{R} g_2^{s_2} \ \wedge\ T_3 = (u_3^{-1} T_2)^{p'} g^{-s_2} g_3^{s_0} \ \wedge\ 1 = T_1^{R} g_1^{-s_1} \\ \quad \wedge\ 1 = T_3^{q'} g_3^{-s_3} \ \wedge\ 1 = T_1^{p'} g_1^{-s_2} \ \wedge\ 1 = T_1^{q'} g_1^{-s_3} \ \wedge\ T_4 = T_1^{p'} g_4^{s_0} \\ \quad \wedge\ T_1^{n} = T_4^{4q'+2} T_1^{2q'+1} g_4^{-4s_3-2s_0} \ \wedge\ T_5 = u_3^{-s_0} u_1^{s_4} u_2^{s_5} g_0^{s_1} \\ \quad \wedge\ 1 = T_1^{z_1} g_1^{-s_4} \ \wedge\ 1 = T_1^{z_2} g_1^{-s_5}\} \end{array}$$

where $CVerC$ uses $s_1 = Rs_0$, $s_2 = s_0 p'$, $s_3 = s_0 q'$, $s_4 = s_0 z_1$, $s_5 = s_0 z_2$, and $CVerV$ checks $T_5 \neq 1$. The four-move concurrent zero-knowledge protocol is straightforward and omitted.

The soundness of our instantiation can be proved in the standard two-stage model [4, 13] for authentications. The zero-knowledge comes from using the standard four-move concurrent zero-knowledge protocol [14, 22]. Random oracles are not used in these proofs. The security reduction is to the DDH assumption in QR_{n^2} for our protocols. Note this assumption is included in Theorem 4's assumptions.

B Proof Sketch of Theorem 4

The DCS scheme $\mathsf{DCS}_{\mathrm{GMR+}}$ is modified from [15]'s by only one step: Including more parameters to the input of $\bar{H}$. Therefore, the proofs of corectnesses, validity, unforgeability, and non-transferability are all similar to that of the proofs in [15]. We omit them here. It remains to prove the invisibility. Note [15] did not even claim the invisibility. There is nothing in their proof to base our proof on. We have to prove our invisibility of the DCS from scratch.

Setup: Let $(\tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^c)$ be the DDH problem instance, where a, b, c are unknown. The simulator $\mathcal{B}$ invokes $CKGS$ and $CKGC$ to set up. $\mathcal{B}$ sets g_1=$\tilde{g}$, $g_2 = \tilde{g}^b$. $\tilde{g}^a$ and $\tilde{g}^c$ will be used in the gauntlet below.

Simulating oracles: $CSig$ is computed using sk_S. $\mathcal{B}$ uses the modified sk_C to compute $CVerC$ as follows: checks σ_3 is a valid signature on $(\sigma - 1, \sigma_2)$ by pk_S; checks $v = u_1^{x_1+\alpha y_1} g_2^{x_2+\alpha y_2}$ where α is computed via Eq. (5), computes R equalling the partial discrete logarithm of $u_3 u_1^{-z_1} u_2^{-z_2}$ with base g_0; checks $u_3 u_1^{-z_1} u_2^{-z_2} = g_0^R$; checks $g^m h^R = \phi$. If all pass indicate to confirm. Else indicate to disavow. Using sk_C, $\mathcal{B}$ can complete the computations of $CVerC$ in conirmation or disavow.

Simulation deviation: It is obviously negligible.

Gauntlet: $\mathcal{A}$ selects a message $m_{ga,1}$. $\mathcal{B}$ selects a random message $m_{ga,0}$, flips a fair coin b, computes the gauntlet DCS $\sigma_{ga} = (\sigma_{ga,1}, \sigma_{ga,2}, \sigma ga, 3)$, $\sigma_{ga,1} = g^{m_{ga,b}} h^{R_{ga}}$, $\sigma_{ga,3} = \mathsf{Sign}(\mathsf{sk}_S, (\sigma_{ga,1}, \sigma_{ga,2})\ )$, $\sigma_{ga,2} = (\hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4)$ where $\hat{u}_1 = \tilde{g}^a$, $\hat{u}_2 = \tilde{g}^c$, $\hat{u}_4 = \hat{u}_1^{x_1+\alpha y_1} \hat{u}_2^{x_2+\alpha y_2}$, $\hat{u}_3 = u_1^{z_1} u_2^{z_2} g_0^{m_{ga,b}}$, α is computed via Eq. (5). The gauntlet DCS is $\sigma_{ga} = (\phi_{ga} = g^{m_{ga,b}} h^{R_{ga}}, \sigma_{ga,2}, \mathsf{Sign}(pk_S, (\phi_{ga}, \sigma_{ga,2})\ )$, where $\sigma_{ga,2} = (\hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4)$.

Extraction: Eventually $\mathcal{A}$ returns $\hat{b}$. $\mathcal{B}$ answers yes to the DDH problem instance if $\hat{b} = 1$. Else, the DDH answer is no.

We need to prove that when (g_1, u_1, g_2, u_2) is not a DDH-yes tuple or $(g_1, u_1, d_3, u_3 g_0^{-R})$, R is the partial discrete logarithm of $u_3 u_1^{-z_1} u_2^{-z_2}$ w.r.t. base g_0, is not a DDH-yes tuple, it is hard for $\mathcal{A}$ to compute in the real world a putative DCS containing these parameters to pass the checkings $u_4 \overset{?}{=} u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$. This can be done by extending Cramer and Shoup [11]'s proof technique of the *smooth projective family* adapted to our new definition of α in Eq. (5). This part of our proof is quite complicated. We leave it to the full version of our paper.

It remains to consider $CVerC$ queries $(m', \sigma', \mathsf{pk}_{S'}, \mathsf{pk}_C)$ that can be transformed from the gauntlet tuple $(m_{ga,b}, \sigma_{ga}, \mathsf{pk}_S, \mathsf{pk}_C)$ that can help $\mathcal{A}$ distinguish b: There are two cases: (Case 1): Inputs to α in Eq.

(5) in the transformed DCS and the gauntlet DCS are identical. Thn a contradiction can be shown that the same method can be used to compute a query to the Cramer-Shoup encryption's decryption oracle and break the Cramer-Shoup encryption. (Case 2): α's inputs are different from the gauntlet DCS to the transformed DCS. Then σ_2 must remain intact in order to be relavant to b. These putative tuples (m', σ', $\mathsf{pk}_{S'}$, pk_C) will result in disavowal by $\mathcal{B}$. $\square$