Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes

Shafi Goldwasser[1,2] and Erez Waisbard[1]

[1] Department of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot 76100, Israel.
{shafi,waisbard}@wisdom.weizmann.ac.il
[2] Laboratory for Computer Science, Massachusetts Institute of Technology.
Cambridge, MA 02139.

Abstract. Since designated confirmer signature schemes were introduced by Chaum and formalized by Okamoto, a number of attempts have been made to design designated confirmer signature schemes which are efficient and at the same time provably secure under standard cryptographic assumptions. Yet, there has been a consistent gap in security claims and analysis between all generic theoretical proposals and any concrete implementation proposal one can envision using in practice. In this paper we propose a modification of Okamoto's definition of security which still captures security against chosen message attack, and yet enables the design of concrete and reasonably efficient designated confirmer signature schemes which can be proved secure without resorting to random oracle assumptions as previously done. In particular, we present simple transformations of the digital signature schemes of Cramer-Shoup, Goldwasser-Micali-Rivest and Gennaro-Halevi-Rabin into secure designated confirmer signature schemes. We prove security of the schemes obtained under the same security assumption made by the digital signature scheme transformed and an encryption scheme we use as a tool.

1 Introduction

Digital signatures introduced by Diffie and Hellman [7] are analogous to signatures in the paper world in the sense that a message that is being signed by the *signer* can later be verified by everyone else. Like in the paper world, a signer can not deny signing a document that carries *his* signature. There are real life scenarios, however, in which the signer wishes that the recipient of the signature would not be able to present the signature to other parties at will.

For example, say a potential employer extends a job offer to a candidate employee including a salary figure. On one hand the employer does not want the employee to show the offer letter to a competitor to elicit a higher salary, and on the other hand the future employee wants to be assured that the offer is binding and can be held up in court. For such a setting we would have like to have a signature schemes in which:

M. Naor (Ed.): TCC 2004, LNCS 2951, pp. 77–100, 2004.

• A court of law (or some other third party) is able (if called upon) to verify the authenticity of the signature.
• No one, but the court of law, should be able to validate the authenticity of the signature (unless the signer steps in).
• The signer should be able to convince the recipient of the signature that it is indeed authentic and can be validated by the court if necessary.

The first attempt to address the issue of signatures that can not be verified by everyone was *Undeniable Signature* by Chaum [5]. Undeniable signatures can not be verified without the signer's cooperation. The signer can either validate a signature or prove it invalid. The problem with this idea is that in any setting where the signer becomes unavailable (of which there may be many) nothing can be determined.

A different idea called *Designated Verifier Signature schemes* was presented by Jakobsson, Sako and Impagliazzo [17]. A designated verifier signature is a signature that can only be validated by a single user, designated by the signer. Designated verifier signatures can used to authenticate the identity of the signer without having the ability to convince any third party of its validity. Its merit is also its weakness. There is indeed no way to force the signer to honor his signature.

Designated Confirmer Signature scheme (DCS), introduced by Chaum [4], address both of the above problems. The parties in a DCS are the signer, the recipient of the signature (aka the verifier) and a *designated confirmer*. The idea of DCS is that during the process of signing, that involves the signer and recipient (as usual), a designated confirmer signature σ is generated. The recipient of the signature cannot convince anyone else of the validity of σ. Rather, the designated confirmer, given σ, has the ability to verify it on his own as well as to convince anyone of its validity/invalidity. The designated confirmer remains completely passive, unless the signer becomes unavailable. In such case, the designated confirmer can either convert the designated confirmer signature into an ordinary signature that can be validated by anyone, or engage in an interactive protocol with any verifier to confirm the validity of the signature. The confirmer is only semi-trusted in the sense that he can only extract/validate signatures for messages which the signer designated him to. Perhaps the most natural candidate to act as a semi-trusted designated confirmer is a court of law.

Going back to the job offer scenario, the employer would sign his offer using a DCS scheme, making the court of law the designated confirmer. Using DCS ensures that the candidate would not be able to convince other employers of the authenticity of the offer, and yet if the employer changes his mind (or becomes unavailable) the candidate can present a signed offer to the court of law and ask for compensation.

A straightforward way to construct a designated confirmer signature scheme, using standard cryptographic primitives, such as public-key encryption scheme and digital signature schemes would be to first sign a message m using an ordinary signature scheme and then encrypt the signature using the designated

confirmer public key[1]. The resulting ciphertext would serve as the designated confirmer signature σ of m. Since the signature is encrypted, only the designated confirmer can be convinced of its validity. Moreover, the designated confirmer can easily extract an ordinary signature from it. One question remains, if the recipient cannot verify the validity of σ on his own, how can he know that he indeed got a valid one? Zero-knowledge naturally comes to the rescue. In order for the recipient to be convinced of the validity of the DCS, the signer and recipient interact in a zero-knowledge proof in which the signer proves to the verifier that what he got is indeed an encryption of a verifiable ordinary signature of m. Since the last assertion is an NP statement, there exist general protocols that achieve this.

The above construction is straight forward and can be easily proved secure. The main problem is that we do not know of *efficient* zero-knowledge proofs for the assertion that the cleartext corresponding to a given ciphertext contains a valid (or invalid) signature of a given document. Proving such statements using general zero-knowledge proofs for NP involve the reduction step to an NP-complete language which makes them unusable in practice. Several works on DCS attempted to remedy the situation and come up with efficient direct DCS constructions. In doing so they either resort to the random oracle assumption for proving security or make no formal claims of security, and thus all trade efficiency with proofs of security in the standard model. We summarize the state of the art in section 1.2.

The **goal of the current paper** is to present DCS schemes with proofs of security in the standard model which do not involve the inefficient step of using general zero-knowledge proofs for proving the validity of signatures.

Our **approach** in achieving this goal is to modify the original definition of security for DCS due to Okamoto [19] to not require zero-knowledge proofs for validity assertions, and then show efficient constructions of DCS schemes which satisfy the new security definition.

We note that an alternative approach toward the same goal would be to construct custom made – tailored to a particular encryption scheme and a particular digital signature scheme – efficient zero-knowledge proofs for the assertion that the cleartext corresponding to a given ciphertext contains a valid (or invalid) signature of a given document. Indeed, utilizing the Cramer-Shoup CCA2 secure public key encryption scheme in some of the confirmation and disavowal protocols proposed in [2] translates to proving statements concerning the equality (and inequality) of discrete logarithms in zero-knowledge. A recent article of Camenisch and Shoup [3] shows ingenious while somewhat complex ways to accomplish this directly without resorting to general zero-knowledge protocols.

[1] All that is required from a designated confirmer in the signing stage is to have a known public key. Other than that the designated confirmer does not need to be aware that his key is used.

1.1 New Results

We propose a new definition of DCS, modifying the original definitions of Okamoto [19] and Camernish and Michels [2] in several ways. The most important modification is to remove the requirement that the confirmation protocols between signer and verifier and confirmer and verifier confirming that a designated confirmer signature is valid must be zero-knowledge[2]. We instead only require that the resulting scheme is existentially unforgeable in the presence of chosen message attack. We stress that a forgery in this context is the ability of anyone but the legal signer to convince a verifier of knowledge of a valid signature of any message. This includes also those messages which have already been signed by the legal signer. Naturally, in the latter case of messages which already have been signed, also the designated confirmer can convince a verifier of the knowledge of valid signatures for these messages, but for no other message.

We give a general transformation that takes any standard digital signature scheme and a public key encryption scheme and turns them into a designated confirmer signature scheme. We prove that if the originating signature scheme is existentially unforgeable under chosen message attack and the public key encryption is secure against CCA2, then the resulting designated confirmer signature scheme is provably secure according to the new definition under the same assumptions made by the digital signature scheme and the encryption scheme.

The main tool our general transformation uses is *strong witness hiding proofs of knowledge* (SWHPOK). Witness hiding proofs of knowledge(WHPOK) for polynomial time verifiable relations R as defined originally by [9], only guarantee that on input x all witnesses w s.t. $(x, w) \in R$ remains hidden. SWHOPK require the additional property that on input x the protocol does not reveal witnesses w' for any other inputs $x' \neq x$. Notably, the general WHPOK protocols for polynomial time verifiable relations [14,9] which exist if one way permutations exist, are already SWHPOK.

Having removed the requirement that the signing and confirmation protocols are zero-knowledge enables using SWHPOK protocols for this purpose instead. The witness in question is a standard digital signature of a message in the sense of [16]. We remark that witness hiding proofs (even strong ones) are in general easier to design than zero-knowledge proofs. Moreover, for a large class of concrete digital signature schemes – including Cramer-Shoup signatures [6], Goldwasser-Micali-Rivest signatures [16] and the Gennaro-Halevi-Rabin signatures [15] – we give simple and direct strong witness hiding proofs of knowledge of a signature for the scheme at hand. Thus, for these digital signature schemes, we give concrete designated confirmer signature schemes which are proved secure under the same

[2] An important implication of removing the indistinguishability security requirement, is that [19] proved that designated confirmer signature scheme and public-key encryption are equivalent. The way Okamoto proved that designated confirmer signature imply public key encryption was based on the indistinguishability between a designated confirmer signature of a message m and a fake signature. He used a valid signature to encrypt the bit 0 and a fake signature to sign the bit 1. Clearly, after modifying the security requirement, this proof no longer holds.

cryptographic assumption the original signature scheme was based on and the existence of a CCA2 secure public key encryption scheme.

The second tool our transformation uses is to take a strong witness hiding proof of knowledge of a signature and modify it so as in the process of proving this knowledge the signer also "encrypts" the signature in the confirmer public key so that the confirmer can later "decrypt" and extract the signature. We prove that if the verifier accepts the proof of knowledge, then with high probability the confirmer will be able to extract the signature from the transcript between signer and verifier. We call this modification of a strong witness hiding proof of knowledge *encrypted strong witness hiding proof of knowledge*. The designated confirmer signature of a message is defined to be this transcript of encrypted proof of knowledge. We use the ideas of Camenisch and Damgard [1] in their work on *verifiable encryption* to get encrypted witness hiding proofs of knowledge.[3]

Putting the above ideas together it is straight forward to get a DCS construction from an standard digital signature scheme. For a message m, the signer first produces an ordinary signature of m, denoted $\sigma(m)$. Next, the signer and verifier engage in a encrypted strong witness hiding proof of knowledge of $\sigma(m)$. If the verifier accepts, the transcript of the interaction can be stored by the verifier as the designated confirmer signature of m. Presented with the transcript, the confirmer can extract $\sigma(m)$ from it, and prove knowledge of $\sigma(m)$ using a strong witness hiding proof of knowledge thus confirming the validity of the designated confirmer signature.

Lastly, we note that unlike the SWHPOK protocols for signing and confirmation of validity of a designated confirmer signature, we still advocate and use in our general transformation a **zero-knowledge** proof for the invalidity of a designated confirmer signature – a so called *Disavowal* protocol. This is natural, as when σ' is an invalid designated confirmer signature, there is no witness to speak of whose secrecy one needs to protect! We argue this has little effect on the overall efficiency of the scheme as we expect to rarely use *Disavowal*. Whereas in *undeniable signature schemes* proving the invalidity of a signature via a *Disavowal* protocol had a crucial role, since it was up to the signer to either confirm or disavow an alleged signature and refusal to disavow could be interpreted as confirming it, this is no longer the case in DCS schemes. The need for disavowal protocol in a DCS scheme arises only when a cheating verifier claims an invalid designated confirmer signature σ' is indeed valid. Since the verifier cannot convince anyone of the signature's validity without the help of the designated confirmer, it usually suffice that the designated confirmer will say

[3] [1] propose an elegant technique of modifying any 3-round honest verifier zero-knowledge proofs for a relation R so that at the end of the protocol the verifier will be guaranteed with high probability to hold a semantically secure encryption of witness w for a given x where $(w, x) \in R$. They showed relevance of this idea to group signatures, signature sharing, and fair exchange of signatures. We note that we apply the [1] transformation to strong witness hiding proofs rather than to zero-knowledge proofs, and thus can not use the claims they prove about the resulting encryption being semantically secure. Still, the resulting protocol can be shown to work in our context as well.

that the verifier is cheating. The need for disavowal protocol may of course arise in the case where the cheating verifier is charged by the law and a proof of his blame needs to be presented. We expect this to rarely occur.

1.2 Related Work

Soon after Chaum introduced the notion of DCS[4], Okamoto presented a formal model and definition of security of DCS and proved (constructively) that secure designated confirmer signature schemes are equivalent to secure public-key encryption [19]. In a nutshell, his definition requires zero-knowledge confirmation protocols of the validity of the signature (or disavowal of its validity) as a way of ensuring non-transferability of the ability to validate a signature. In addition to theoretical results, Okamoto also gives two concrete practical schemes without an argument nor claim of security. Indeed, [18] showed that one of Okamoto's schemes enables the designated confirmer to universally forge signatures.

Michels and Stadler [18] suggest how to use a tool called *designated confirmer commitments* to construct designated confirmer signature scheme starting from any Fiat-Shamir like signature scheme [11] The resulting DCS schemes can be proved secure only in the random oracle model, inheriting this property from the use of the Fiat-Shamir paradigm for constructing signatures. Another DCS scheme suggested in [18] is based on deterministic RSA signatures which are existentially forgeable and thus again, unless one resorts to the use of the "hash then sign" techniques which are provably secure in the random oracle model. [2] point out attacks on previous DCS schemes (including [18]) when several signers share the same confirmer. They strengthen the DCS security requirements of [19] to address these problems, and show the existence of a secure DCS (under the new definition) using general tools of existentially unforgeable digital signatures schemes, CCA2 secure encryption schemes, and general concurrent ZK protocols for NP statements. For this definition [2] propose concrete implementations of DCS based on either deterministic RSA signatures (or Fiat-Shamir like signatures) whose security again is provable in the random oracle model. Some of the confirmation and disavowal protocols proposed in [2] when using the Cramer-Shoup public encryption function as the underlying CCA2 secure encryption amount to proving statements concerning the equality (and inequality) of discrete logarithms in zero-knowledge. A recent article of Camenisch and Shoup [3] shows direct ways to accomplish this.

2 New Definition for a DCS

2.1 Informal Outline of the Definition

The model consists of three players: signer S, verifier V and designated confirmer C. Throughout, *all parties* receive as input the public keys of the signer and of the designated confirmer, denoted by PK_s and PK_c. The signer has an auxiliary secret input, denoted SK_s and the confirmer has an auxiliary secret input, denoted SK_c.

A pair of important algorithms with respect to which validity of a designated confirmer signature is defined are ***Extract*** and ***Verify***. On inputs a message m, a designated confirmer signature σ, PK_s, PK_c, and SK_c, algorithm Extract either outputs $fail$ or a string σ^*, which can be publicly verified as a valid ordinary digital signature of m with respect to PK_s(as defined in [16]) by running the verification algorithm Verify. In essence, Extract turns a designated confirmer signature that can be verified only by a confirmer into an ordinary digital signature that can be validated by anyone.

The definition also calls for the existence of three main interactive protocols:

ConfirmedSign: a *protocol* between the *signer* and a *verifier* on a common input message m, which produces as output either an accept or reject vote by the verifier along with a string σ referred to as the **designated confirmer signature** of m. If the verifier accept then the string σ should be a valid designated confirmer signature, that is one that can be transformed to an ordinary digital signature using the Extract algorithm. Here the verifier is the recipient of the designated confirmer signature that needs to be convinced of its validity.

By combining the signing process along with the confirmation process we deviate from the definition of [19,18,2]. We argue that this is a natural modification, as the recipient of a DCS always needs to be convinced of the validity of the DCS, thus in practice, the two actions are always performed together.

Conf: a *protocol* between the *confirmer* and a *verifier* on common input a message m and a designated confirmer signature σ, at the end of which the verifier either accept or reject σ as a valid designated confirmer signature of m. If σ is a *valid* designated confirmer signature (i.e. one from which the Extract algorithm can output an ordinary valid signature of m) then the confirmer should be able to convince the verifier of its validity. Here the verifier can be *any* party that needs to be convinced of the validity of the DCS.

Disavowal: a *protocol* between the *confirmer* and *verifier* on common input a message m and a designated confirmer signature σ, at the end of which the verifier either accepts or rejects σ as an *invalid* designated confirmer signature of m (where an invalid designated signature σ is one for which ***Extract*** outputs $fail$). As in $Conf$, the verifier can be *any* party that needs to be convinced of the validity of the DCS.

The **security requirements** we make fall into two categories: security for signers and security for the confirmer,

1. **Security for the signer:** For any message m *not previously signed*, no one, except for the legal signer can
   a) Run $ConfirmedSign(m, \cdots)$ in the role of the prover, successfully with non-negligible probability.
   b) Produce a publicly verifiable ordinary signature σ^* of m with respect to the signer's public signing key (i.e. $Verify(, PK_s, m, \sigma^*) =$ valid).
   c) Produce a designated confirmer signature σ for m which the legal designated confirmer will confirm as valid with non-negligible probability.

For any *previously signed* message m, no one, except for the legal signer and the legal designated confirmer, can do 1a, 1b as above.[4]

2. **Security for the confirmer:** No one but the legal signer S and designated confirmer C, including any coalition of signers $\{S_j\}$ where all $S_j \neq S$ sharing the same confirmer, can confirm a designated confirmer signature for a message previously signed with respect to SK_s.

2.2 Formal Definition

In the coming definition, $negl(k)$ denotes any function which grows slower than $\frac{1}{k^c}$ for all c for all k sufficiently large.

Definition 1 *A* secure designated confirmer signature scheme *consists of the following components:*

1. ***Key Generation Algorithms*** (G_s, G_c)*:* G_c *is a probabilistic polynomial time algorithm that on input* 1^n *(where* n *is the security parameter), outputs a pair of strings* (SK_c, PK_c) *(the designated confirmer's private and public key respectively).* G_s *is a probabilistic polynomial time algorithm that on input* 1^n*, outputs a pair of strings* (SK_s, PK_s) *(the signer's private and public key respectively).*
2. ***Signature Extraction****: A pair of polynomial time algorithms* $(Extract, Verify)$ *such that* $Extract$ *on inputs* m, σ, PK_s, PK_c *and* SK_c *returns a string* σ^* *and* $Verify$ *on input* PK_s, m *and* σ^* *outputs valid or invalid. If* $Verify(PK_s, m, \sigma^*) = valid$*, where* $\sigma^* = Extract(m, \sigma, PK_s, PK_c, SK_c)$*, then we say that the* $Extract$ *algorithm was successful and* σ^* *is a valid ordinary signature of* m *with respect to* PK_s*.*
3. ***ConfirmedSign****: An interactive protocol referred to as* $ConfirmedSign_{(S,V)}$ *between interactive probabilistic polynomial time algorithms(ITM)* S *and* V *which on common inputs* (m, PK_s, PK_c) *outputs a pair* (b, σ) *where* $b \in \{accept, reject\}$ *and* σ *is refereed to as the designated confirmer signature of a signer* S *on message* m*. The requirements from* $ConfirmedSign$ *are:* $\exists$ V *such that*
   a) **completeness:** $\exists$ S *(with auxiliary input* SK_s*), such that* $\forall$ $(PK_s, SK_s) \in G_s$ *and* $(PK_c, SK_c) \in G_c$*,* $\forall$ m*,* $ConfirmedSign_{(S,V)}(m, PK_s, PK_c)$ *outputs* $(accept, \sigma)$ *such that* $Verify(PK_s, m, Extract(m, \sigma, PK_s, PK_c, SK_c)) = valid$*.*
   b) **Soundness:** $\forall$ *probabilistic polynomial time* S' *with auxiliary input*[5] y*,* $\forall$ m $Pr[ConfirmedSign_{(S',V)}(m, PK_s.PK_c)$ *outputs* $(accept, \sigma)$

[4] Ordinary signature schemes are secure if a forger cannot produce a signature on a message that has not been signed before. It is not required that a forger would not be able to produce a different signature on previously signed messages. Similarly, for a designated confirmer signature scheme, we do not require (in part 6 of Def 1) that it is infeasible for a forger F to produce *new* valid designated confirmer signatures for messages previously signed.

[5] This y captures the possible history, available to attackers, of interaction with Signers, Confirmers and Verifiers.

such that $Verify(PK_s, m, Extract(m, \sigma, PK_s, PK_c, SK_c)) \neq valid] < negl(n)$
The probability is taken over all possible coins of the key generation algorithms G_s, G_c *and algorithms* S', V, *and* $Extract$.

4. ***Confirmation*** *An interactive protocol referred to as* $Conf_{(C,V)}$ *between interactive probabilistic polynomial time algorithms(ITM)* C *and* V *which on common inputs* (m, σ, PK_s, PK_c) *outputs* $b \in \{accept, reject\}$. *The requirements from* $Conf$ *are:* $\exists$ V *such that*

   a) **Completeness:** $\exists$ C *(with auxiliary input* SK_C*), such that*
   $\forall$ $(PK_s, SK_s) \in G_s$ *and* $(PK_c, SK_c) \in G_c$, $\forall$ m, *if*
   $Verify(PK_s, m, Extract(m, \sigma, PK_s, PK_c, SK_c) = valid$
   then $Conf_{(C,V)}(m, \sigma, PK_s, PK_c)$ *outputs accept*

   b) **Soundness:** $\forall$ *probabilistic polynomial time* C' *(with auxiliary input* y*)*
   if $Verify(PK_s, m, Extract(m, \sigma, PK_s, PK_c, SK_c) \neq valid$
   then $Pr(Confirm_{(C',V)}(m, \sigma, PK_s, PK_c)$ *outputs accept*$) < negl(n)$
   The probability is taken over all possible coins of C', V, $Extract$ *and the key generation algorithms* G_s *and* G_c.

5. ***Disavowal*** *An interactive protocol referred to as* $Disvowal_{(C,V)}$ *between interactive probabilistic polynomial time algorithms(ITM)* C *and* V *which on common inputs* (m, σ, PK_s, PK_c) *outputs* $b \in \{accept, reject\}$, *The requirements from* $Disvowal$ *are:* $\exists$ V *such that*

   a) **Completeness:** $\exists$ C *(with auxiliary input* SK_c*), such that*
   $\forall$ $(PK_s, SK_s) \in G_s$ *and* $(PK_c, SK_c) \in G_c$, $\forall$ m, *if*
   $Verify(PK_s, m, Extract(m, \sigma, PK_s, PK_c, SK_c) \neq valid$,
   then $Disvowal_{(C,V)}(m, \sigma, PK_s, PK_c)$ *outputs accept*

   b) **Soundness:** $\forall$ *probabilistic polynomial-time* C' *(with auxiliary input* y*),*
   if $Verify(PK_s, m, Extract(m, \sigma, PK_s, PK_c, SK_c)) = valid$
   then $Pr(Disvowal_{(C',V)}(m, \sigma, PK_s, PK_c)$ *outputs accept*$) < negl(n)$
   The probability is taken over all possible coins of C', V, $Extract$ *and the key generation algorithms* G_s *and* G_c.

6. *We say that a designated confirmer signature scheme is secure if it meets the following requirements:*

   a) *Let* F *be a probabilistic polynomial time forging algorithm which, on input strings* PK_s, PK_c *can first request the execution of* $ConfirmedSign_{(S,F)}$, $Conf_{(C,F)}$ *and* $Disvowal_{(C,F)}$ *for polynomially many adaptively chosen inputs of its choice; and then attempts to run the* $ConfirmedSign$ *protocol on* m *of its choice in the role of the prover. We require that for all such* F *and* m

$$Pr(ConfirmedSign_{(F,V)}(1^n, m, PK_s, PK_c) = (accept, \sigma)) < negl(n)$$

   The probability is taken over all possible coins used by F, S, C, V *and the key generation algorithms* G_s *and* G_c.

b) Let F be a probabilistic polynomial time forging algorithm which, on input strings 1^n, PK_s, PK_c can first request the execution of $ConfirmedSign_{(S,F)}$, $Conf_{(C,F)}$ and $Disvowal_{(C,F)}$ for polynomially many adaptively chosen inputs of its choice; and then outputs a pair (m, σ^*). We require that for all such F and m,

$$Pr(Verify(PK_s, m, \sigma^*) = valid) < negl(n)$$

The probability is taken over all possible coins used by F, S, C, V and the key generation algorithms G_s and G_c

c) Let F be a probabilistic polynomial time forging algorithm which, on input strings 1^n, PK_s, PK_c, and SK_c, can first request the execution of $ConfirmedSign_{(S,F)}$ for polynomially many adaptively chosen messages $\{m_i\}$, as well as request the execution of $Conf_{(C,F)}$ and $Disvowal_{(C,F)}$ for polynomially many adaptively chosen inputs; and then outputs a pair (m, σ'). We require that for all such F and for message $m \notin \{m_i\}$ (i.e not previously signed)

$$Pr(Conf_{(C,V)}(1^n, m, \sigma', PK_s, PK_c) = accept) < negl(n)$$

The probability is taken over all possible coins used by F, S, C, V and the key generation algorithms G_s and G_c.[6]

*d) **Security for designated confirmers:** Let A be a probabilistic polynomial time attacking algorithm which, on input strings $1^n, PK_s, PK_c$ can request the execution of $ConfirmedSign_{(S,A)}$, $Conf_{(C,A)}$ and $Disvowal_{(C,A)}$ for polynomially many inputs of his choice and finally, for a pair $\{m, \sigma\}$ of his choice, A executes $Conf_{(A,V)}(1^n, m, \sigma, PK_s, PK_c)$. For all such A*

$$Pr(Conf_{(A,V)}(1^n, m, \sigma, PK_s, PK_c) = accept) < negl(n)$$

The probability is taken over all possible coins used by A, S, C, V and the key generation algorithms G_s and G_c.
Moreover, this should hold when many signers share the same confirmer. Meaning, when A knows polynomially many SK_{s_j} such that $SK_{s_j} \neq SK_S$.

3 Tools

Our transformation uses several tools, including ordinary digital signatures secure against adaptive chosen message attack as defined in [16], public key encryption secure against adaptive chosen ciphertext attack(CCA2) as defined in

[6] Note that this requirement also implies that with high probability even the designated confirmer C can not run successfully $ConfirmedSign_{(F,V)}$ protocol on messages not previously signed, nor produce a valid ordinary signature of a message not previously signed.

[8], canonical strong witness hiding proofs of knowledge defined in subsection 3.1, and encrypted strong witness hiding proofs of knowledge defined in subsection 3.2.

3.1 Strong Witness Hiding Proofs of Knowledge

Witness Hiding proof of knowledge (WHPOK) were defined by Feige and Shamir in [10] as follows.

Let R be a polynomial time relation. Namely, there exists a polynomial p and a polynomial time computable function f such that

$$R = \{(x, w) \text{ such that } f(x, w) = 1, |w| < p(|x|)\}$$

Let $w(x)$ denote the set of w such that $(x, w) \in R$.

Definition 2 ([10]) *: We say that G is a instance generator for relation R if on input 1^n it produces instances $(x, w) \in R$ of length n. We say that G is a* hard instance generator *if for any polynomial time witness finding F, $P[(x, F(x)) \in R] < negl(n)$, where $x \in G(1^n)$. The probability is taken over the coin tosses of G and F.*

Definition 3 ([10]) *: Let (P,V) be a proof of knowledge (POK) system for relation R and let G be a hard instance generator for R. We say that (P,V) is* witness hiding proof of knowledge *(WHPOK) on (R,G) if for any probabilistic polynomial time V' there exist an expected polynomial time witness extractor M, $P[(P(w),V')(x) \in w(x)] < P[M(x) \in w(x)] + negl(n)$ where $x \in G(1^n)$. The probability is taken over the distribution of the inputs chosen by G and witnesses as well as the random tosses of P,V' and M.*

In our context, the relation R we shall be interested in will be the pairs of *a message* and *a valid ordinary digital signature of message*, for some given ordinary digital signature scheme which is secure against chosen message attack. As such, we shall need to deviate from the original definition of WHPOK in a few aspects. First, for a secure digital signature scheme as defined by [16] it is impossible to find a single valid $(message, signature)$ pair in polynomial time for messages not previously signed. Thus, our proofs of knowledge should be witness hiding for any polynomial time distribution over R. Second, we require the proof of knowledge to remain witness hiding, even the verifier chooses the input message to run the protocol on after it participated in many executions of the protocol on different input messages which were chosen adaptively by the verifier himself. We call the modified definition *strong witness hiding proofs of knowledge* (SWHPOK).

Definition 4 *Let (P,V) be a proof of knowledge (POK) system for relation R. We say that (P,V) is* strong witness hiding *(SWH) on R if for any probabilistic polynomial time V' who can in a preliminary stage choose (adaptively) polynomially many x_i and run $(P(w_i),V')(x_i)$, and only later choose a challenge $x \neq x_i$,*

there exist a witness extractor M which runs in expected polynomial time such that $P[(P(w), V')(x) \in w(x)] < P[M(x) \in w(x)] + negl(n)$. The probability is taken over the distribution of witnesses as well as random coin tosses of P, V', G and M.

Finally, in order to be able to apply the technique of Camenisch and Damgard [1] of encrypted witness hiding proofs of knowledge, we require that our protocols will be of canonical form defined as follows.

Definition 5 *A* Canonical witness hiding proof of knowledge *for a boolean relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is a three-move SWHPOK for R which is defined by three probabilistic procedures $(P_1, P_3, Verdict)$, satisfying the following conditions:*
1. On common input x and an auxiliary input w, P's first step uses P_1 to compute the first message to be sent t and some side information r. At the second step the verifier sends a random bit string c as a challenge. At the third step the prover uses x, w, r and c as input to P_3 to compute a response s, which he sends to the verifier. In the forth step the verifier uses the predicate $Verdict$ taking x, t, c and s as inputs to check whether s is a valid response. A triple (t, c, s), such that $Verdict(x, t, c, s)$ accepts is called an accepting triple for x.
2. The number of possible challenges that can be sent by the verifier is polynomial in the security parameter.
3. There exist a knowledge extractor that can extract the witness from knowing the answer to all possible challenges.

Note that we have added the requirement of being strong WH into what we call a canonical WHPOK. Also, note that there is no requirement above of having negligible soundness probability. Indeed in all the canonical WHPOK that on which we perform transformations in this paper, the challenge of the verifier is a single bit which yields an overall soundness probability of $\frac{1}{2}$.

3.2 Encrypted Strong Witness Hiding Proofs of Knowledge of a Signature

An important tool used by the construction is called a *encrypted strong witness hiding proof of knowledge*. The idea is as in [1] where they apply the technique to zero-knowledge protocols.

Start with any signature scheme existentially unforgeable under adaptive chosen message attack $\Sigma = (SG, Sign, Verify)$ where SG is the key generation algorithm, and $Sign$ (and $Verify$) are the signing (and verifying) algorithms. Define the relation

$$R_\Sigma = \{((PK_s, m), \sigma) : Verify(PK_s, m, \sigma) = valid, (PK_s, SK_s) \in SG(1^k)\}$$

Assume you are given, for simplicity of exposition, a canonical strong WH-POK for relation R_Σ defined by three probabilistic algorithms $(P_1, P_2, Verdict)$ where the number of possible challenges of the verifier is two and the soundness

probability is $\frac{1}{2}$ (in general the construction works for any polynomial number of challenges).

Canonical witness hiding proof of knowledge for R_Σ:
Common input to both prover and verifier is (PK_s, m). Auxiliary input to the prover is σ, such that $((PK_s, m), \sigma) \in R_\Sigma$.

1. The prover computes $(t, r) = P_1((PK_s, m), \sigma)$ and sends t to the verifier.
2. The verifier selects $b \in_R \{0, 1\}$ and send it to the prover.
3. The prover calculates $s = P_3((PK_s, m), \sigma, r, b)$ and sends it to the verifier.
4. The verifier accepts if $Verdict((PK_s, m), t, b, s) = 1$, otherwise he rejects.

Now, let $Enc = (EG, E, D)$ be a given a CCA2 secure public key encryption scheme. In our context, the public encryption key (and corresponding secret decryption key) will be of the designated confirmer C and we denote them by PK_c (and SK_c respectively). The above protocol is turned into an *encrypted witness hiding proof of knowledge for* R_Σ as follows.

Encrypted canonical witness hiding proof of knowledge for R_Σ:
Common input to both prover and verifier is m, PK_s, PK_c. Auxiliary input to the prover is σ such that $((PK_s, m), \sigma) \in R_\Sigma$.

1. The prover computes $(t, r) = P_1((PK_s, m), \sigma)$, $s_0 = P_3((PK_s, m), \sigma, r, 0)$ and $s_1 = P_3((PK_s, m), \sigma, r, 1)$; encrypts s_0 and s_1, using the designated confirmer's public key to obtains $e_0 \in E_{PK_c}(s_0)$ and $e_1 \in E_{PK_c}(s_1)$. Then, the prover sends (t, e_0, e_1) to the verifier.
2. The verifier selects $b \in_R \{0, 1\}$ and sends it to the prover.
3. The prover reveals s_b and the random coins r_b that were used in the encryption.
4. If $E_{PK_c}(s_b, r_b) = e_b$ and $Verdict((PK_s, m), t, b, s_b) = 1$, the verifier accepts, otherwise he rejects.

Essentially, in this protocol at the first round the prover sends an encrypted answer to both possible challenges one of which will be decrypted on the third round.

Running this basic protocol k times in sequence decreases the probability of cheating to $\frac{1}{2^k}$, but costs a possibly prohibitive $3k$ rounds. To reduce the the number of rounds to constant maintaining negligible probability of error, we can employ ideas similar to Goldreich-Kahan[13][7] or utilize a trapdoor commitment scheme as suggested in [1].

[7] Recalling, the idea of [13], is to simply add an initial round in which the verifier commits to all his challenges in advance $b_1, \cdots b_k$, followed by k parallel executions of the above 3-round protocol with the modification that the verifier decommits its challenges $b_1, \cdots, b_k$ in step (2) rather than simply sending them in the clear. This transformation maintains the SWH property.

Theorem 6 *The modified protocol remains a canonical (strong) witness hiding proof of knowledge for the relation* R_Σ

4 A General Construction of Designated Confirmer Signature

Let $\Sigma = (SG, Sign, Verify)$ be a signature scheme which is existentially unforgeable under chosen message attack which has a canonical strong WHPOK for the relation R_Σ[8] . Let $Enc = (EG, E, D)$ be a CCA2 secure encryption scheme.

In the following we let S denote the signer, V the verifier in the various protocols, and C the designated confirmer. We let $(PK_s, SK_s) \in SG(1^k)$ denote the public verification key and the secret signing key of the signer and $(PK_c, SK_c) \in EG(1^k)$ be the public encryption key and the private decryption key of the designated confirmer.

The Designated Confirmer signature scheme:

Key Generation Algorithms: $(G_s, G_c) = (SG, EG)$. The key generation algorithm consists of the pair of key generation algorithm for the signature and encryption schemes in use.

ConfirmedSign protocol: S computes an ordinary signature σ of m by computing $\sigma \in Sign_{SK_s}(m)$. Then, the encrypted canonical witness hiding proof of knowledge for R_Σ of section 3.2 is run between the signer S in the role of a prover and verifier V on common inputs m, PK_s and PK_c and auxiliary input σ to S.

The triple $\sigma' = (t, e_0, e_1)$ (defined during the protocol) is defined to be the designated confirmer signature of message m with respect to PK_s. When the protocol is repeated k times, the designated confirmer signature of m is $\{(t_i, e_{i0}, e_{i1}), i = 1, \cdots, k\}$.

Signature Extraction: Extracting an ordinary signature σ of message m such that
$Verify(PK_S, m, \sigma) = valid$ from a designated confirmer signature σ' where $(accept, \sigma') \in ConfirmedSign_{S,V}(m, PK_s, PK_c)$ is straightforward for C. C simply decrypts e_0 and e_1 to obtain s_0 and s_1. Knowing both s_0 and s_1 implies extraction of the ordinary signature σ using the knowledge extractor of the witness hiding proof of knowledge protocol for σ.[9]

[8] At the moment we are assuming we are given such a WHPOK. We know that such WHPOK exist if one-way permutations exist. Later we will show efficient construction of such protocols for a large family of signature schemes.

[9] Note that the probability that upon decryption it is discovered that s_0, s_1 were not properly encrypted is essentially the same as the soundness probability of the witness hiding protocol. In the full protocol with k repetitions where the designated confirmer signature is (t_i, e_{i_0}, e_{i_1}) for $i = 1, \cdots, k$, the probability that there exist a pair e_{i_0}, e_{i_1} which properly decrypts to a pair s_0 and s_1 from which σ can be decrypted is negligibly close to 1.

Confirmation protocol: On common inputs m, an alleged designated confirmer signature σ', and PK_s, PK_c the following protocol is run between confirmer C and verifier V. C has as an auxiliary input SK_c. First, C extracts an ordinary signature of m, by $\sigma = Extract(m, \sigma', PK_s, PK_c, SK_c)$. If $Verify(PK_s, m, \sigma) = invalid$, then the confirmation protocol outputs invalid and stops. If $Verify(PK_s, m, \sigma) = valid$, then C (as prover) and V (as verifier) run the canonical strong WHPOK for R_Σ on common input (PK_s, m) and auxiliary input σ to C.

Disavowal protocol $Disavowal_{(C,V)}$**:** Given m and alleged DCS σ' for which $Verify(PK_s, m, Extract(m, \sigma, PK_s, PK_c, SK_c)) = invalid$ the disavowal protocol is a zero-knowledge proof that $Verify(PK_s, m, Extract(m, \sigma, PK_s, PK_c, SK_c)) = invalid$.
The latter is obviously an NP statement.

Theorem 7 *The above system constitutes a secure* Designated Confirmer Signature scheme*, given that Sign is existentially unforgeable under chosen message attack and Enc is a CCA2 secure public key encryption scheme*

Proof. For brevity, let us include only a sketch of the proof.

First, we need to show that any polynomial time adversary A, participating in $ConfirmedSign_{(S,A)}$, $Conf_{(C,A)}$ and $Disavowal_{(C,A)}$ in the role of the verifier on polynomially many messages $m_1, \ldots, m_k$ of his choice, cannot successfully run $ConfirmedSign_{(A,V)}(m, PK_s, PK_c)$ or compute an ordinary signature σ^* such that $Verify(PK_s, m, \sigma^*) = valid$ for any message m of his choice (regardless whether $m \in \{m_1, \cdots, m_k\}$ or not). Suppose for contradiction that such an A does exist. Since $ConfirmedSign_{(A,V)}(m, PK_s, PK_c)$ is a proof of knowledge, an adversary A that successfully run $ConfirmedSign_{(A,V)}(m, PK_s, PK_c)$, can also extract an ordinary signature σ' of m with high probability. This contradicts the assumption that $ConfirmedSign_{(S,A)}$ and $Conf_{(C,A)}$ are *strong witness hiding* and thus do not reveal an ordinary signature for any message.

Next, we need to show that such F cannot produce a pair (m, σ) where $Conf_{(C,V)}(m, \sigma, PK_s, PK_c)$ will be successful, namely for which $Verify(PK_s, m, Extract(m, \sigma, PK_s, PK_c, SK_c)) = valid$, for a new m not previously signed. Suppose for contradiction that such an A does exist. Then, a success of A would constitute a successful malleability attack on the encryption scheme E_{PK_c} which is impossible as E_{PK_c} was taken to be secure against CCA2.

Finally, we need to show that any coalition of probabilistic polynomial time adversaries $\{A_i\}$ with secret signing keys $\{SK_i\}$, playing $ConfirmedSign_{(S,A_i)}$, $Conf_{(C,A_i)}$ and $Disavowal_{(C,A_i)}$ in the role of the verifier on polynomially many messages $m_1, \ldots, m_k$ of their choice, cannot successfully run $Conf_{(A_i,V)}$ on any pair (m, σ) . Here again, since $Conf_{(A_i,V)}$ is a POK, successfully running $Conf_{(A_i,V)}$, means that A_i can extract an ordinary signature of m with high probability which contradicts the fact that $Conf$ and $SignedConf$ are witness hiding. □

4.1 On the Complexity of the Construction

Unlike the efficient WHPOK $ConfirmedSign$ and $Conf$, the *Disavowal* protocol is a less efficient ZKPOK. We claim that due to the rare expected use of *Disavowal* it has very lite effect on the overall efficiency of the scheme. See discussion in 1.1.

One problematic point is that per our description the verifier must store the designated confirmer signature in its entirety, i.e $(t, e_{i_0}, e_{i_1}), 1 \leq i \leq k$, in case it needs to be presented in a later time to the confirmer for confirmation. If the signer was honest, σ can be extracted from any of the triples (t, e_{i_0}, e_{i_1}) and thus saving a single triplet would significantly reduce the storage needed. However, saving a single triplet does not suffice in case of a cheating signer as it may be triple which does not enable extraction and was not detected during the signing protocol with probability $\frac{1}{2}$. By choosing to store only a random subset of the triples (hoping you store at least one proper one), one may tradeoff the probability of being able to eventually extract and storage.[10]

5 Cramer-Shoup Based DCS

In this section we show how to transform the Cramer-Shoup (CS) signature scheme [6] into a designated confirmer signature scheme. Since the CS signature scheme is existentially unforgeable under chosen message attack, using the construction in 4 we can transform it into a DCS scheme. In order to do that we describe a canonical WHPOK of a CS signature.

5.1 The Cramer-Shoup Signature Scheme

The Cramer-Shoup signature scheme [6] is an efficient signature scheme, which is existentially unforgeable under chosen message attack under the strong RSA assumption.

Definition 8 *The* **strong RSA assumption** *is the assumption that given a randomly chosen RSA modulus n and a random $z \in Z_n^*$, it is hard find $r > 1$ and $y \in Z_n^*$, such that $y^r = z$.*

The Cramer-Shoup scheme:

Key Generation: Two random l'-bit primes p and q are chosen, where $p = 2p' + 1$ and $q = 2q' + 1$, with both p' and q' prime. Let $N = pq$. Also chosen are $h, x \in QR_N$ and a random $(l+1)-$bit prime e'. The private key is (p, q) and the public key is (N, h, x, e') along with a collision resistance hash function H (e.g. SHA-1).

Signature generation: To sign a message m, a random $(l+1)$ bit prime $e \neq e'$ is chosen and a random $x' \in QR_N$ is chosen The equation $y^e = xh^{H(x')} mod\ N$

[10] A back of the envelope calculation shows that if one chooses at random l out of k pairs to store, the probability (after having passed the confirmation protocol) that the confirmer will not be able to extract the signature is $\frac{1}{2^l} \frac{1}{(k-l)^l}$.

is solved for y and the equation $(y')^{e'} = x'h^{H(m)} mod\ N$ is solved for y'. The Cramer-Shoup signature is (e, y, y').

Signature verification: To verify a signature (e, y, y') on a message m, e is first checked to be an odd $(l+1)$-bit number different from e'. Second, $x' = (y')^{e'} h^{-H(m)} mod\ N$ is computed. Third, it is checked that $x = y^e h^{-H(x')} mod\ N$.

5.2 Canonical WHPOK of a CS Signature

Proving knowledge of a CS signature of a message m amounts to proving knowledge of (e, y, y') such that $\exists e, x'$ satisfying the equations $y^e = xh^{H(x')}\ mod\ N$ and $(y')^{e'} = x'h^{H(m)}\ mod\ N$. In order to prove knowledge of a CS signature we use a ZKPOK of the ith root as a tool.

Protocol I: Zero-knowledge proof of knowledge of the ith root:

On common input w, i, N such that $w = s^i \bmod N$, and auxiliary secret input s to the prover.

1. The prover picks $r \in_R Z_n^*$, computes $v = r^i\ mod\ N$ and sends v to the verifier.
2. The verifier picks $b \in_R \{0, 1\}$ and sends b to the prover.
3. The prover sends $t = rs^b\ mod\ N$ to the verifier.
4. The verifier accepts iff $t^i \equiv vw^b\ (mod\ N)$. (To achieve lower soundness probability the protocol may be repeated.)

Theorem 9 *Protocol I is a perfect zero-knowledge proof of knowledge of s.*

Protocol II: Strong WHPOK of Cramer-Shoup signatures. On common input message m, a Cramer-Shoup public key (N, h, x, e') and an auxiliary secret input to the prover (e, y, y') (a Cramer-Shoup signature of m).

1. The prover sends e, x' to the verifier where $x' = (y')^{e'} h^{-H(m)}\ mod\ N$.
2. The prover proves in zero-knowledge (using Protocol I of 3.2) that he knows a y, such that $y^e = xh^{H(x')}\ mod\ N$ and that he knows a y', such that $(y')^{e'} = x'h^{H(m)}\ mod\ N$.

Theorem 10 *Protocol II is a strong WHPOK of a Cramer-Shoup signatures*

Proof. It is easy to see that **completeness** holds - a prover that knows a CS signature of m can always convince a verifier.

Since we are using the ZKPOK of a modular root, there exist a knowledge extractor for y (the eth root of $xh^{H(x')}$) and y'(the e'th root of $x'h^{H(m)}$). These y and y', together with the e given in the first round are a CS signature of m, hence a **witness-extractor** exist.

Soundness holds because a cheating prover, that does not know a CS signature, cannot prove knowledge of either, the eth root of $xh^{H(x')}$, or the e'th root of $x'h^{H(m)}$. Thus, the soundness is guaranteed by the soundness of the POK of the eth root.

The most important thing we need to prove in order to apply the general construction to the above protocol is that it is indeed **strong witness hiding**. It was already proved in [6] that seeing a Cramer-Shoup signature on polynomially many messages does not enable an adversary to sign any **new** message that has not been signed before, let alone seeing only a partial CS signature. It remains to show that executing the above protocol does not reveal the Cramer-Shoup signature of any of the messages on which it was run. Assume toward contradiction that there exist an adversary A that on a Cramer-Shoup public key (N, h, x, e'), executes the above protocol in the role of a verifier with the signer in the role of a prover on polynomially many messages of the verifiers choice $m_1, \ldots, m_t$ and finally outputs, with non-negligible probability, a pair (m, σ), where $m \in \{m_1, \ldots, m_t\}$ and σ is a valid Cramer-Shoup signature of m. We show that such algorithm A can be used to construct the following forging algorithm B for the standard Cramer-Shoup signature scheme. B will utilize A's algorithm for this purpose (i.e B will run A on different inputs and random tapes).

The Forging Algorithm B:

Algorithm B's input: A Cramer-Shoup public key (N, h, x, e') and access to A's program.
Algorithm B's output: A pair (m, σ), where m is a message and $\sigma = (e, y, y')$ is a Cramer-Shoup signature of m.

1. **Query phase**: Initially B interacts with A where B acts in the role of the prover and A the verifier in protocol II above, perfectly simulating A's view of interacting with legitimate signer without ever querying the signer. On message m_i of A's choice, B proves to A that he knows a Cramer-Shoup signature of m_i in the following way:
   a) B picks a random $(l+1)$-bit prime e_i and $x'_i \in_R QR_N$ and sends (e_i, x'_i) to A.
   b) B proves to A in zero-knowledge that he knows y_i, such that $y_i^{e_i} = xh^{H(x'_i)} \ mod \ N$ and y'_i, such that $(y'_i)^{e'} = x'_i h^{H(m_i)} \ mod \ N$. Naturally, B does not know such y_i and y'_i. Nevertheless, B can perfectly simulate A's view using the standard rewinding technique for proving zero-knowledge - taking advantage on the ability to rewind A upon a challenge that B was not prepared for[11]
2. **Output phase**: If A outputs a valid Cramer-Shoup signature σ for $m \in \{m_1, \ldots m_t\}$ (or any m for that matter), then B outputs $< m, \sigma >$.

Clearly, B runs in expected polynomial time as so does A. B perfectly simulates A's view as the x'_i and e_i are uniformly distributed (completely independent of the m_i) and thus if A outputs a Cramer-Shoup signature with non-negligible

[11] The number of possible challenges in each round of the ZKPOK of the e'th root is 2 and thus running the protocol simultaneously for y and y' brings the number of possible challenges to 4 and can be easily simulated.

probability, so does B, contradicting the fact that the Cramer-Shoup signature scheme is existentially unforgeable under the strong RSA assumption.

□

We remark that one could simplify protocol II further and rather than running step 2 as it is, allow the verifier to choose at random whether to engage in a WHPOK of y such that $y^e = xh^{H(x')} \ mod \ N$ (step 2(a) in protocol II), or a WHPOK of y', such that $(y')^{e'} = x'h^{H(m)} \ mod \ N$ (step 2(b) in protocol II) but not both. Since, knowing a legal Cramer-Shoup signature of m means knowing both y and y', a cheating prover who cannot answer both challenges will be caught with probability $\frac{1}{2}$.

Finally, protocol II did not have a canonical form. It can be easily turned canonical 3-round protocol (to be repeated in turn k times), included for completion.

Canonical strong WHPOK of Cramer-Shoup signature m: On common input m, Cramer-Shoup public key (N, h, x, e') and auxiliary secret input to prover (e, y, y').

1. prover calculates $x' = (y')^{e'}h^{-H(m)} \ mod \ N$, picks $r, r' \in_R Z_n^*$, computes $v = r^e \ mod \ N$, $v' = r'^{e'} \ mod \ N$ and sends e, x', v, v' to the verifier.
2. verifier picks $b, b' \in_R \{0, 1\}$ and them to the prover.
3. prover sends $t = ry^b \ mod \ N$ and $t' = r'y'^{b'} \ mod \ N$ to the verifier.
4. verifier accepts iff $t^e \equiv v(xh^{H(x')})^b \ (mod \ N)$ and $t'^{e'} \equiv v'(x'h^{H(m)})^{b'} \ (mod \ N)$.

6 Goldwasser-Micali-Rivest Based DCS

In this section we show how to transform the Goldwasser-Micali-Rivest (GMR) signature scheme into a designated confirmer signature scheme. Since the GMR signature scheme is existentially unforgeable under chosen message attack, using the construction in 4 we can transform it into a DCS scheme. In order to do that we describe a canonical strong WHPOK of a GMR signature.

6.1 The GMR Signature Scheme

The digital signature scheme of Golwasser Micali and Rivest [16] is existentially unforgeable under chosen message attack under the assumption that claw-free trapdoor permutation (pairs f_0, f_1 for which it is hard to find x, y such that $f_0(x) = f_1(y)$) exist. In [16] it is shown that such family of trapdoor permutation exists if factoring is hard.

Before we describe the scheme we recall the followings notation:

Definition 11 *Let* $\sigma = \sigma_1\sigma_2 \dots \sigma_n$ *where* $\sigma_i \in \{0, 1\}$. *we denote by* $f_\sigma(x) = f_{\sigma_1}(f_{\sigma_2}(\cdots f_{\sigma_n}(x)\cdots))$ *and* $f_\sigma^{-1}(y) = f_{\sigma_n}^{-1}(f_{\sigma_{n-1}}^{-1}(\cdots f_{\sigma_1}^{-1}(y)\cdots))$

The GMR scheme is defined by the following three probabilistic algorithms:

Key Generation: Choose two pairs of claw-free permutations, (f_0, f_1) from a common domain D_f and (g_0, g_1), from a common domain D_g for which you know $f_0^{-1}, f_1^{-1}, g_0^{-1}, g_1^{-1}$. Uniformly choose $X \in D_f$. The public key is: $(D_f, X, f_0, f_1, g_0, g_1)$ and the secret key is $(f_0^{-1}, f_1^{-1}, g_0^{-1}, g_1^{-1})$.

Signing a message: We denote by H the history and we set $H_1 = \phi$. To sign the ith message m_i, uniformly choose $R_i \in D_g$. Set $z_1^i = f_{H_i \circ R_i}^{-1}(X)$ and $z_2^i = g_{m_i}^{-1}(R_i)$. The signature is $\sigma(m_i) = (z_1^i, z_2^i, H_i)$ and the history is updated, setting $H_{i+1} = H_i \circ R_i$.

Verifying a signature (z_1, z_2, H)**:** Accept iff $f_{H \circ R}(z_1) = X$ for $R = g_m(z_2)$.

Theorem 12 ([16]) *: If claw-free permutations exist, the above scheme is existentially unforgeable under chosen message attack.*

6.2 Factoring Based GMR Scheme

An implementation based on intractability assumption of factoring is suggested in [16]. Let $N = pq$ be the product of two primes satisfying $p \equiv q \equiv 3(mod\ 4)$ and $p \neq q(mod\ 8)$. $f_0 = x^2\ mod\ N$ and $f_1 = 4x^2\ mod\ N$ are permutations over the set of quadratic residues mod N.

Theorem 13 ([16]) *Under the intractability assumption of factoring the $(f_0, f_1)-$pair are claw-free trapdoor permutations.*

It was noted by Goldreich [12] that the factoring based implementation of the GMR can be sped up. For the $(f_0, f_1)-$pair described above, a fast way of computing $f_\alpha^{-1}(x)$, where $|\alpha| = k$, is by computing

$$f_\alpha^{-1}(x) = \frac{R_N(2^k, x)}{(R_N(2^k, 4))^{i(\alpha)}}$$

Where $i(\alpha)$ denotes the integer encoding of α and $R_N(2^k, x)$ denotes the 2^kth root of x modulo N.

6.3 Canonical WHPOK of the Factoring Based GMR Signature

Proving knowledge of a GMR signature amounts to proving knowledge of a triple (z_1, z_2, H_i) such that $\exists R_i$, such that $z_2 = g_m^{-1}(R_i)$ and $z_1 = f_{H_i \circ R_i}^{-1}(X)$. The WHPOK of a GMR signature that we present[12] takes advantage on the special structure of the factoring based GMR scheme. Let $f_0(x) = x^2\ mod\ N_1$, $f_1(x) = 4x^2\ mod\ N_1$, $g_0(x) = x^2\ mod\ N_2$ and $g_1(x) = 4x^2\ mod\ N_2$. Our

[12] In[16] a tree like structure is imposed on the H_i's, but here, for simplicity, we discuss the simpler and less efficient version in which the H_i's grows linearly in the number of signed messages.

protocol uses the fact that in the factoring based GMR scheme, proving knowledge of $g_m^{-1}(R_i)$ and $f_{H_i \circ R_i}^{-1}(X)$ is done by proving knowledge of modular roots. Thus, we can use the ZKPOK of the ith root from 3.2 as a tool, toward a canonical WHPOK of a GMR signature of m.

Protocol III: Strong WHPOK of a Factoring based GMR signature : On a common input m and public key $(N_1, N_2, X \in Z_{N_1}^*)$ and an auxiliary input to the prover $\sigma = (z_1, z_2, H_i)$ (a valid GMR-signature of m).

1. The prover computes $R_i = g_m(z_2)$ and sends R_i, H_i to the verifier.
2. The prover proves in zero-knowledge that he knows (z_1, z_2) such that $z_2 = g_m^{-1}(R_i)$ and $z_1 = f_{H_i \circ R_i}^{-1}(X)$. Proving knowledge of $g_m^{-1}(R_i)$ amounts to proving knowledge of the $2^{|m|}$th root of $R_i \pmod{N_2}$ and proving knowledge of the $2^{i(|m|)}$th root of 4 $\pmod{N_2}$. Namely, knowing how to calculate both the nominator and the denominator in $g_m^{-1}(R_i) = \frac{R_N(2^{|m|}, R_i)}{(R_N(2^{|m|}, 4))^{i(m)}}$.
   Similarly, proving knowledge of $f_{H_i \circ R_i}^{-1}(X)$ amounts to proving knowledge of the $2^{|H_i \circ R_i|}$th root of $X \pmod{N_1}$ and proving knowledge of the $2^{i(|H_i \circ R_i|)}$th root of 4 $\pmod{N_1}$

Theorem 14 *Protocol III is a strong WHPOK of a GMR signature of m.*

Proof. The proof is essentially the same as 5.2. We include it for completion. It is easy to see that **completeness** holds - a prover that knows a GMR signature of m can always convince a verifier.

Since we are using the ZKPOK of a modular root, there exist a knowledge extractor for the $2^{|m|}$th root of $R_i \pmod{N_2}$ and the $2^{i(|m|)}$th root of 4, hence there exist a knowledge extractor for $z_2 = g_m^{-1}(R_i)$. Similarly there exist a knowledge extractor for $z_1 = f_{H_i \circ R_i}^{-1}(X)$. These z_1, z_2, together with the H_i given in the first round are a GMR signature of m, hence a **witness-extractor** exist.

Soundness holds because a cheating prover, that does not know a GMR signature, cannot prove knowledge of at least one of the modular roots he is required to in step 2 of the protocol of 6.3. Thus, a cheating prover has a probability at most $\frac{3}{4}$ to fool the verifier. Repeating step 2 k times this probability is reduced to $(\frac{3}{4})^k$.

We now show that the above protocol is **strong witness hiding**. It was already proved in [16] that seeing GMR signatures for $m_1, \ldots m_t$ chosen adaptively by the adversary does not enable an adversary to produce a GMR signature for a new $m \notin \{m_1, \ldots m_t\}$, let alone seeing partial GMR signatures. But, suppose toward contradiction that there exists an adversary A, which after running the above protocol III on message $m_1, \ldots m_t$ adaptively chosen can produce a GMR signature (z_1, z_2, H_i) for an $m \in \{m_1, \ldots m_t\}$. We show that the existence of such A implies that the original GMR scheme is not existentially unforgeable and thus contradicts the existence of claw-free trapdoor permutations assumption (e.g. factoring is hard).

Intuitively, since R_i and H_i are chosen at random, independently from the message m and the public key, they do not allow an adversary to sign a message. Formally, using A as an internal procedure whose inputs and random tape can be set, we describe an algorithm B that on a GMR public key forges GMR signatures.

Algorithm B's input: GMR public verifying key $PK = (N_1, N_2, X \in Z^*_{N_1})$ and A's program.
Algorithm B's output: pair (m, σ) where σ is a valid GMR-signature of m with respect to PK.

1. Initially, B runs algorithm A on input PK. For each chosen message m_i by A, B proves to A on common inputs (m_i, PK) that he knows a GMR signature of m_i (as in protocol III) as follows .
   a) B chooses $R_i \in_R D_g$ and $H_i \in_R D_g$ and gives R_i, H_i to A.
   b) B proves in zero-knowledge that he knows z_1, z_2 such that $z_2 = g_m^{-1}(R_i)$ and $z_1 = h^{-1}_{H_i \circ R_i}(X)$. Naturally, B does not know such z_1 and z_2, nevertheless, B can perfectly simulate A's view using the standard rewinding technique for proving zero-knowledge - taking advantage on the ability to rewind A upon a challenge that B was not prepared for.
2. If A outputs a valid GMR signature (z_1, z_2, H) for $m \in \{m_1, \ldots m_t\}$, then B outputs $(m, (z_1, z_2, H))$.

Clearly, B runs in probabilistic polynomial time as does A. In step 1(c) B perfectly simulates A's view and thus in step 2, the adversary A would output a GMR signature with the same probability as when running with the true signer. □

We remark that one could simplify the above protocol III further (as we did in the Cramer-Shoup case) so that the verifier chooses at random whether the prover will prove knowledge of z_1 s.t. $z_1 = h^{-1}_{H_i \circ R_i}(X)$, or knowledge of z_2 s.t. $z_2 = g_m^{-1}(R_i)$, but not both.

7 Gennaro-Halevi-Rabin Based DCS

In this section we show how to transform the Gennaro-Halevi-Rabin digital signature (denoted the GHR scheme) [15] into a designated confirmer signature scheme. The GHR-signature scheme is existentially unforgeable under chosen message attack, assuming the strong RSA assumption.

The idea of GHR-signatures is as follows. Let the public key be a triple (n, h, x) where n is an RSA modulus, $x \in_R Z^*_n$ and $h \in_R H$ where H is family of hash functions which [15] is proved to exists under the strong RSA assumption. On a message m, the signature is defined to be $\sigma_n(m) = x^{\frac{1}{h(m)}} \bmod n$.

7.1 Transforming GHR Signatures into a DCS Scheme

In order to turn the GHR signature scheme into a designated confirmer signature scheme using the type of ideas we have used in this paper, we need to give a canonical WHPOK of $R = \{(m, \sigma_n(m))\}$ to be used as a confirmation protocol between signer and verifier. In fact, we do better than that and can give a 3-round zero-knowledge proof of knowledge of a signature.

ZKPOK of a GHR signature of message m: On common input message m and public-key (n, h, x), and a prover's auxiliary input a signature of m, $x^{\frac{1}{h(m)}} \bmod n$:

1. The prover picks a random $r' \in Z_n^*$ and calculates $r = (r')^{h(m)} \bmod n$ (which makes $r' \equiv r^{\frac{1}{h(m)}} \ (mod\ n)$) and sends r to the verifier.
2. The verifier picks $b \in_R \{0, 1\}$ and sends it to the signer.
3. The prover sends $c = r'(x^{\frac{1}{h(m)}})^b \ mod\ n$ to the verifier.
4. The verifier accepts iff $c^{h(m)} \equiv rx^b \ (mod\ n)$.

The above protocol is repeated k times and the verifier accepts iff he accepts in each of the iterations, dropping the error probability to $\frac{1}{2^k}$. It is easy to verify that the protocol is ZKPOK (with respect to sequential repetitions) with standard methods (similarly to the proof given in 3.2). Using the Goldreich-Kahan [13] methods it can be converted to constant rounds.

7.2 Transforming the Deterministic RSA into a DCS Scheme

Instead of using the GHR-signature scheme and the strong-RSA assumption, we could use an even simpler version of the above protocols to get a DCS scheme starting from the plain RSA scheme itself[20]. Let (n, e) be the RSA public key and d be the RSA secret exponent. A RSA signature of m is $m^d \ mod\ n$. Thus, proving knowledge of RSA signature of m amounts to proving knowledge of the dth modular root of m. This can be done using the ZKPOK of the ith modular root that we already described in 3.2.

We note that as RSA itself is existentially forgeable, so will be the DCS originating from it. Interestingly, however, whereas the plain RSA scheme is universally forgeable under chosen message attack, this is no longer true for the deterministic RSA based DCS. The reason is that the verifier can no longer request the signer for RSA signatures of messages of his choice, but only to execute $ConfirmedSign_{(S,V)}$ (where the signer proves knowledge of an ordinary RSA signature without revealing it). Thus, in a sense the DCS obtained by transforming the RSA signature scheme into a DCS scheme is more secure than the signature scheme one starts with.

Acknowledgements. We are grateful to O. Goldreich, A. Shamir, M. Naor and A. Kipnis for their useful comments. We also like to thank the anonymous referees for their useful and detailed comments.

References

1. J. Camenisch, Ivan Damgård. Verifiable Encryption, Group Encryption, and Their Applications to Separable Group Signatures and Signature Sharing Schemes. ASIACRYPT 2000 pp. 331–345
2. J. Camenisch, M. Michels. Confirmer Signature Schemes Secure against Adaptive Adversaries. EUROCRYPT 2000 pp. 243–258
3. J. Camenisch, V. Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. Cryptology ePrint Archive (November 2002).
4. D. Chaum. Designated confirmer signatures. In EUROCRYPT '94, vol. 950 of LNCS, pp. 86–91. Springer Verlag, 1994.
5. D. Chaum, H. Van Antwerpen. Undeniable Signatures. In CRYPTO 1989 pp. 212–216
6. R. Cramer, V. Shoup. Signature Schemes Based on the Strong RSA Assumption. ACM Conference on Computer and Communications Security 1999 pp. 46–51
7. W. Diffie and M. Hellman, New directions in cryptography. IEEE Trans. Inform. Theory IT-22, (Nov. 1976), pp. 644–654.
8. D. Dolev, C. Dwork, M. Naor. Non-Malleable Cryptography (Extended Abstract). STOC 1991 pp. 542–552
9. U. Feige, A. Fiat, A. Shamir. Zero-Knowledge Proofs of Identity. Journal of Cryptology 1(2) pp. 77–94 (1988)
10. U. Feige, A. Shamir: Witness Indistinguishable and Witness Hiding Protocols. STOC 1990. pp. 416–426.
11. A. Fiat, A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. CRYPTO 1986. pp. 186–194
12. O. Goldreich. Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme. CRYPTO 1986 pp. 104–110
13. O. Goldreich, A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP, Journal of Cryptology 1995.
14. O. Goldreich, S. Micali, A. Wigderson. How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design. CRYPTO 1986: 171–185
15. R. Gennaro, S. Halevi, T. Rabin. Secure Hash-and-Sign Signatures Without the Random Oracle. EUROCRYPT 1999 pp. 123–139
16. S. Goldwasser, S. Micali, R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. 17(2): 281–308 (1988)
17. M. Jakobsson, K. Sako, R. Impagliazzo. Designated Verifier Proofs and Their Applications. EUROCRYPT 1996 pp. 143–154
18. M. Michels, M. Stadler. Generic Constructions for Secure and Efficient Confirmer Signature Schemes. EUROCRYPT 1998: 406–421
19. T. Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In CRYPTO '94, vol. 839 of LNCS, pp. 61–74. Springer Verlag, 1994.
20. R. L. Rivest, A. Shamir, L. M. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. CACM 21(2): 120–126 (1978)