A Note on a Convertible Undeniable Signature Scheme with Delegatable Verification

Jacob C. N. Schuldt and Kanta Matsuura

Institute of Industrial Science, University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan.
{schuldt, kanta}@iis.u-tokyo.ac.jp

Abstract. Undeniable signatures, introduced by Chaum and van Antwerpen, require a verifier to interact with the signer to verify a signature, and hence allow the signer to control the verifiability of his signatures. Convertible undeniable signatures, introduced by Boyar, Chaum, Damgård, and Pedersen, furthermore allow the signer to convert signatures to publicly verifiable ones by publicizing a verification token, either for individual signatures or for all signatures universally. In addition, the signer is able to delegate the ability to prove validity and convert signatures to a semi-trusted third party by providing a verification key. While the latter functionality is implemented by the early convertible undeniable signature schemes, the recent schemes do not consider this despite its practical appeal.
In this note we present an updated definition and security model for schemes allowing delegation, and highlight a security property, token soundness, which is often implicitly assumed but not formally treated in the description of the security model for convertible undeniable signatures. We also note that the straightforward implementations of the efficient convertible undeniable signature schemes recently proposed by Phong, Kurosawa and Ogata do not allow a verifier to check the correctness of a public key, which essentially allows a malicious signer to break the token soundness of the schemes. We then propose a convertible undeniable signature scheme inspired by the recent designated confirmer signature scheme by Schuldt and Matsuura. The scheme allows delegation of verification, does not require verifiers to hold public/private key pairs, and is provably secure in the standard model assuming the computational co-Diffie-Hellman problem, a closely related problem, and the decisional linear problem are hard. Compared to the most efficient scheme by Phong *et al.*, our scheme has slightly larger signatures, but allows delegation of verification, is based on arguably more natural security assumptions, and has significantly shorter tokens for individual conversion of signatures. Lastly, our scheme does not require tokens to be verifier specific or the presence of a trusted third party, which seems to be needed to guarantee the token soundness of the schemes by Phong *et al.*

Keywords: undeniable signatures, universal/selective convertibility, provable security

1 Introduction

Undeniable signatures, first introduced by Chaum and van Antwerpen [12], are like ordinary signatures, except that verification of a signature requires interaction with the signer. Unlike ordinary signatures, this enables a signer to control who can verify his signatures and when verification is allowed. This feature makes undeniable signatures attractive when sensitive data or confidential business agreements are being signed, since the signer is guaranteed that only the relevant parties can verify his signature and no outsider will be able to link him to the signed data. To preserve non-repudiation, an undeniable signature scheme furthermore requires that a signer is able to disavow an invalid signature. Hence, a signer will either be able to confirm or disavow the validity of any signature, and any dispute can be resolved by letting the signer convince a judge about the validity or invalidity of the signature in question. Since their introduction, a number of undeniable signature schemes have been proposed e.g. [11, 18, 31, 26, 24, 32, 23].

In [7], Boyar, Chaum, Damgård and Pedersen introduced convertible undeniable signatures which allow the signer to convert his undeniable signatures into publicly verifiable signatures. Two types of conversions were introduced: *selective conversion* which enables the signer to individually convert signatures, and *universal conversion* which enables the signer to convert all (existing and future) signatures. A signer selectively converts a signature σ on a message m by releasing a token tk_σ which will convince any verifier that σ is indeed a valid signature on m. Likewise, a signer universally converts all signatures by releasing a universal token tk_* which can be used as a token for any signature. This feature is desirable when public verifiability is required after a period of time, which, for example, is the case for the problem of keeping digital records of confidential political decision (e.g. see [16]). Another aspect of the definition in [7] is that the private key material of the signer is divided into two parts: a signing key sk and a verification key vk. The former is only used to sign messages, whereas the latter is used to convert and confirm or disavow signatures. This property is useful in scenarios where the signer is met with more verification requests than he has capacity to handle, or the signer might become off-line or otherwise unavailable and therefor cannot handle verification requests. In such scenarios, the signer will be able to delegate the verification by releasing vk to a semi-trusted entity who will then have the capacity to verify signatures on behalf of the signer. It is required that the scheme remains unforgeable, even for the semi-trusted entity with the knowledge of vk.

The original scheme by Boyar *et al.* [7] was shown to be insecure by Michels, Petersen and Horster [28] when an universal token is released. Michels *et al.* furthermore proposed an updated scheme, but only heuristic arguments for the security of this scheme were presented. Recently, Aimani and Vergnaud [3] provided an analysis of the updated scheme in the generic group model. Furthermore, Damgård and Pedersen [16] proposed two convertible undeniable signature schemes based on El Gamal signatures, but did not give full proofs of invisibility, and Michels and Stadler [29] proposed a scheme based on Schnorr signatures.

The first RSA based scheme was proposed by Gennaro, Rabin and Krawczyk [18] which Miyazaki later improved [30] (see also [19]). Kurosawa and Takagi [25] proposed a (selective convertible only) scheme which they claimed to be the first RSA based scheme secure in the standard model, but it was shown by Phong, Kurosawa and Ogata [34] that the scheme does not provide full invisibility. Phong *et al.* furthermore proposed a new selective and universally convertible RSA based scheme secure in the standard model.

Laguillaumie and Vergnaud [27] defined and proposed a pairing-based time-selective convertible undeniable signatures which allow conversion of signatures constructed in a given time period, and Monnerat and Vaudenay [31] pointed out that their efficient MOVA undeniable signature scheme supports selective conversion although a formal analysis is not given. Recently, Huang, Mu, Susilo and Wu [22] proposed the currently most efficient scheme in the random oracle model which supports both selective and universal conversion. Yuen, Au, Liu and Susilo [38] proposed a selective and universal convertible standard model scheme, but it was shown by Phong, Kurosawa and Ogata [33] that the scheme is not invisible for the standard definition of invisibility. Phong *et al.* furthermore proposed two efficient schemes which are claimed to be the first practical discrete logarithm based schemes both providing selective and universal conversion and being provably secure in the standard model.

An intuitive approach to the construction of a convertible undeniable signature scheme is to use an encryption scheme to encrypt (parts of) an ordinary signature, and this is indeed the approach used in [16, 33]. The challenge in these type of schemes is to provide efficient confirm and disavow protocols as well as reasonable short conversion tokens. Aimani [1, 2] proposed a generic construction based on a certain class of encryption and signature schemes. However, this approach does not provide selective conversion as described above; while a signer is able to extract a valid public verifiable signature from an undeniable signature, a verifier will not receive any proof that the received publicly verifiable signature corresponds to the undeniable signature i.e. the verifier does not receive a token which allows him to independently verify the undeniable signature, but only a publicly verifiable signature derived from the undeniable signature. It should be noted that a designated confirmer signature, in which the signer holds both signer and confirmer key pairs, will not automatically yield a selective and universal convertible signature scheme for a similar reason; the ability of the confirmer to extract a publicly verifiable signature from an undeniable signature does not necessarily imply the ability to provide a token which will convince a verifier of the validity of the original signature[1].

All of the early proposed schemes [7, 28, 16, 18] implement the above described separation of the signer's key material into a signer key and verification key, which allows delegation of the verification. However, despite the practical advantages of this property, it is not considered in the formalization, security model or the concrete schemes presented in the recent papers [27, 25, 22, 38, 1, 34, 33]. In these schemes, only the signer is able to confirm, disavow and convert signatures and no mechanism is provided for delegating this ability[2]. Note that although the possession of an universal token allows verification of any signature, this does not necessarily provide the ability to efficiently prove validity to a third party in a non-transferable way.

In this note. We present an updated definition and security model for schemes allowing delegation, and highlight a security property, token soundness, which is often implicitly assumed but not formally treated in the description of the used security models[3]. Token soundness guarantee that a malicious signer (or delegated verifier) cannot produce a token such that an invalid signature/message pair appears valid. This is different from the ordinary completeness requirement which only considers honestly generated tokens, and will furthermore, in combination with unforgeability of undeniable signatures, guarantee unforgeability of message/signature/token tuples (see Section 4). We also note that the straightforward implementations of the efficient schemes by Phong, Kurosawa and Ogata [33] do not allow a verifier to check the correctness of a public key, which essentially allows a malicious signer to break the token soundness of the scheme.

We then propose a convertible undeniable signature scheme which allows verification delegation and is provably secure in the standard model assuming the computational co-Diffie-Hellman problem, a closely related problem, and the decisional linear problem are hard (see Section 2). The scheme is based on the basic designated confirmer scheme recently proposed by Schuldt and Matsuura [36]. Unlike [36], verifiers are not required to hold public/private

[1] While the use of NIZK proofs might seem natural when considering token generation, this approach is also somewhat problematic in the standard model since a common reference string is required, and this cannot be generated by the signer due to a conflict between the zero-knowledge property of the NIZK and the required token soundness of the scheme (see also the comments about the schemes by Phong *et al.* [33] in Section 4).

[2] We note that while the random oracle model schemes in [27, 22] do not consider delegation of verification, it only requires knowledge of part of the private signer key to run confirm, disavow and convert. Defining this part as the verification key and replacing the universal token with this, seems to provide schemes satisfying our definition of verification delegation. While this would require the security to be re-proved (since the adversary gains access to parts of the private signer key in some scenarios), we believe that this is possible.

[3] We note that [27] and [21] briefly mention soundness of the verification algorithms as a requirement, but many other recent papers, e.g. [22, 38, 1, 34, 33], do not explicitly describe this property.

key pairs, key registration is not needed, and a token generation method is provided. We highlight that while the aim of the constructions in [36] is to provide on-line non-transferability for designated confirmer schemes, the proposed scheme only aims at providing "standard" off-line non-transferability.

Besides enabling delegation of verification, our scheme has properties which compare favorably to those of the efficient schemes by Phong *et al.* [33]. Firstly, our scheme relies on arguably more natural assumption compared to the schemes in [33] which require the q-strong Diffie-Hellman assumption. Secondly, the tokens used for selective conversion in our scheme correspond to two group elements or roughly $2 \cdot 170$ bits for approximately 80 bits of security, whereas the schemes in [33] requires tokens of size $13 \cdot 170$ bits. Lastly, our scheme does not require tokens to be verifier specific or the use of a trusted third party, which seems to be needed to guarantee the token soundness in the schemes from [33]. We note that the signature size of our scheme is identical to that of the second scheme in [33], while being roughly 100 bits larger than that of the first scheme.

2 Preliminaries

Negligible function. A function $\epsilon : \mathbb{N} \rightarrow [0,1]$ is said to be *negligible* if for all $c > 0$ there exists an n_c such that for all $n > n_c$ $e(n) < 1/n^c$.

Bilinear maps. Our scheme makes use of groups equipped with a *bilinear map* (we refer the reader to [5] for a detailed description of these maps). To instantiate our schemes, we consider a generator $\mathcal{G}$ that on input 1^k outputs a description of groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p where $2^k < p < 2^{k+1}$, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and an isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$. We will use the notation $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi)$ as a shorthand for the output of $\mathcal{G}$.

The discrete logarithm problem in $\mathbb{G}_2$. Given $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi)$, and random elements $g_2, h \in \mathbb{G}_2$, the discrete logarithm problem in $\mathbb{G}_2$ is defined as computing $x \in \mathbb{Z}_p$ such that $g_2^x = h$. We say that the discrete logarithm problem is *hard* in $\mathbb{G}_2$ if all polynomial time algorithms have negligible probability (in the parameter k) of solving the problem.

The computational co-Diffie-Hellman problem in $(\mathbb{G}_1, \mathbb{G}_2)$. Given $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi)$, elements $g_2, g_2^a \in \mathbb{G}_2$ and $h \in \mathbb{G}_1$ where a is a random element in $\mathbb{Z}_p$, the computational co-Diffie-Hellman problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is to compute $h^a \in \mathbb{G}_1$. We say the computational co-Diffie-Hellman problem is *hard* in $(\mathbb{G}_1, \mathbb{G}_2)$ if all polynomial time algorithms have negligible probability (in the parameter k) of solving the problem.

Besides the "standard" computational co-Diffie-Hellman problem defined above, we will also consider the following variant: Given $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi)$, elements $g_2, g_2^a, h \in \mathbb{G}_2$ where a is a random element in $\mathbb{Z}_p$, compute $\psi(h^a) \in \mathbb{G}_1$. We will refer to this problem as the computational ψ-Diffie-Hellman problem to distinguish it from the above, and say that the problem is *hard* in $(\mathbb{G}_1, \mathbb{G}_2)$ if all polynomial time algorithms have negligible probability (in the parameter k) of solving the problem.

The decisional linear problem in $\mathbb{G}_2$. Given $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi)$ and elements $u, v, u^x, v^y, h, h^z \in \mathbb{G}_2$ where x, y are random elements in $\mathbb{Z}_p$, the decisional linear problem in $\mathbb{G}_2$ is to decide whether $z = x + y$ or z is a random element in $\mathbb{Z}_p$. We say the decisional linear problem is *hard* in $\mathbb{G}_2$ if all polynomial time algorithms have negligible probability (in the parameter k) of solving the the problem.

Sigma protocols. A sigma protocol for a binary relation R is a 3-move protocol between a prover and a verifier. Both prover and verifier receive a common input x, but the prover receives a witness y such that $(x, y) \in R$ as an additional private input. In the first move of the protocol, the prover sends a "commitment" message a, in the second move, the verifier sends a random "challenge" message c, and in the final move, the prover sends a "response" message z. Given the response message, the verifier either accepts or rejects the proof. A sigma protocol is required to have two security properties:

- *Special honest verifier zero-knowledge:* There exists a simulation algorithm $\mathtt{Sim}_{\Sigma}$ that given input x and a challenge message c, outputs an accepting transcript $(a, c, z) \leftarrow \mathtt{Sim}_{\Sigma}(x, c)$. We require that the simulated (a, c, z) is perfectly indistinguishable from the transcript of a real interaction, conditioned on the event that the verifier chooses c as his challenge message.
- *Special soundness:* There exists an algorithm $\mathtt{WExt}_{\Sigma}$ that, given two accepting transcripts, (a, c, z) and (a, c', z'), for input x which have the same commitment message a but different challenge messages $c \neq c'$, can extract a witness y such that $(x, y) \in R$.

In the construction of our schemes, we will make use of sigma protocols for proving various relations among discrete logarithms, and we will use the notation

$$\Sigma\{(x, y) : g^x = v \wedge u^x w^y = e\}$$

to mean a sigma protocol in which the prover receives the private input (x, y) and proves to the verifier that the equations $g^x = v$ and $u^x w^y = e$ holds for group elements $g, v \in \mathbb{G}$ and $u, w, e \in \mathbb{G}'$ where $\mathbb{G}$ and $\mathbb{G}'$ might be different groups of the same order. Such sigma protocols are relatively straightforward to construct using the well-known protocol for proving knowledge of a discrete logarithm by Schnorr [35] as a building block, and we refer the reader to the analysis by Camenisch and Stadler [10] for more details.

3 Convertible Undeniable Signatures

A convertible undeniable signature (CUS) scheme consists of the following algorithms and protocols:

- `Setup`: Given input 1^k, this algorithm outputs a set of public parameters par.
- `KeyGen`: Given par, this algorithm outputs a public key pk, a private verification key vk and a private signing key sk. The verification key vk will be used as private prover input in the confirm/disavow protocols, and to selectively convert signatures, whereas the signing key will only be used to sign messages.
- `Sign`: Given par, sk and a message m, this algorithm outputs an undeniable signature σ.
- `Convert`: Given par, vk and (m, σ), this algorithm returns a verification token tk_σ for σ if (m, σ) is a valid message/signature pair. Otherwise, the algorithm returns $\perp$.
- `Verify`: Given par, pk, (m, σ) and tk_σ, this algorithm returns either `accept` or `reject`.
- $(\mathtt{Confirm}, \mathtt{V}_{con})$: A pair of interactive algorithms for confirming validity of a signature. Both algorithms take as common input par, pk and (m, σ). The algorithm `Confirm` takes as an additional private input the verification key vk. While `Confirm` has no local output, the algorithm $\mathtt{V}_{con}$ will output either `accept` or `reject` after having interacted with `Confirm`.

```
$Exp^{\texttt{sound}}_{S,\mathcal{A}}(1^k)$
  $par \leftarrow \texttt{Setup}(1^k)$
  $(pk^*, m^*, \sigma^*, st) \leftarrow \mathcal{A}(par)$
  if $pk^* \notin \mathcal{PK}$
    output 0
  $z_1 \leftarrow_2 \{A(st) \leftrightarrow V_{con}\}(par, pk^*, m^*, \sigma^*)$
  $z_2 \leftarrow_2 \{A(st) \leftrightarrow V_{dis}\}(par, pk^*, m^*, \sigma^*)$
  if $z_1 = z_2 = \texttt{accept}$
    output 1
  else output 0
```

```
$Exp^{\texttt{tk-sound}}_{S,\mathcal{A}}(1^k)$
  $par \leftarrow \texttt{Setup}(1^k)$
  $(pk^*, m^*, \sigma^*, tk^*_\sigma, st) \leftarrow \mathcal{A}(par)$
  if $pk^* \notin \mathcal{PK}$
    output 0
  $z \leftarrow_2 \{A(st) \leftrightarrow V_{dis}\}(par, pk^*, m^*, \sigma^*)$
  if $\texttt{Verify}(par, pk^*, m^*, \sigma^*, tk^*_\sigma) = \texttt{accept} \wedge$
    $z = \texttt{accept}$
    output 1
  else output 0
```

Fig. 1. Soundness experiments

– $(\texttt{Disavow}, \texttt{V}_{dis})$: A pair of interactive algorithms for disavowing validity of a signature. Similar to the above, both algorithms take as common input par, pk and (m, σ), and the algorithm `Disavow` takes the verification key vk as an additional private input. `Disavow` has no local output, but $\texttt{V}_{dis}$ will output either `accept` or `reject` after having interacted with `Disavow`.

Note that the above definition does not explicitly mention how a universal token is generated or how signatures are verified using this token since this functionality follows directly from the separation of the private key material into a signing key sk and a verification key vk. More precisely, a universal token corresponds to vk, and using this, any signature can be verified using the `Convert` algorithm. We assume that given a public key pk, it is possible to decide if pk belongs to the set $\mathcal{PK}$ of valid public keys (i.e. $\mathcal{PK}$ is the set of all possible public keys output by `KeyGen`). Furthermore, we also assume that given (pk, vk) it can be verified that vk is a verification key for the public key pk.

We use the notation $\{\texttt{Confirm}(vk) \leftrightarrow \texttt{V}_{con}\}(par, pk, m, \sigma)$ to denote the interaction between `Confirm` and $\texttt{V}_{con}$ with the common input (par, pk, m, σ) and the private input vk to the `Confirm` algorithm (a similar notation is used for `Disavow` and $\texttt{V}_{dis}$). We furthermore use $z \leftarrow_2 \{\texttt{Confirm}(vk) \leftrightarrow \texttt{V}_{con}\}(par, pk, m, \sigma)$ to denote the output of V_{con} upon completion of the protocol i.e. z will be either `accept` or `reject`.

Correctness. It is required that for all parameters $par \leftarrow \texttt{Setup}(1^k)$, all keys $(pk, vk, sk) \leftarrow \texttt{KeyGen}(par)$, all messages $m \in \{0,1\}^*$, and all signatures $\sigma \leftarrow \texttt{Sign}(par, sk, m)$, that the interaction $z \leftarrow_2 \{\texttt{Confirm}(vk) \leftrightarrow \texttt{V}_{con}\}(par, pk, m, \sigma)$ yields $z = \texttt{accept}$. Furthermore, it is required that for all $tk_\sigma \leftarrow \texttt{Convert}(par, vk, m, \sigma)$ that $\texttt{Verify}(par, pk, m, \sigma, tk_\sigma) = \texttt{accept}$. Lastly, it is required that for any $(m', \sigma') \notin \{(m, \sigma) : \sigma \leftarrow \texttt{Sign}(par, sk, m)\}$, the interaction $z' \leftarrow_2 \{\texttt{Disavow}(vk) \leftrightarrow \texttt{V}_{dis}\}(par, pk, m', \sigma')\}$ yields $z' = \texttt{accept}$.

4 Security

4.1 Soundness

Soundness of a CUS scheme S is defined via the experiment $Exp^{\texttt{sound}}_{S,\mathcal{A}}(1^k)$ shown in Figure 1. In the experiment, we define the advantage of the algorithm $\mathcal{A}$ as

$$\texttt{Adv}^{\texttt{sound}}_{S,\mathcal{A}} = \Pr[Exp^{\texttt{sound}}_{S,\mathcal{A}}(1^k) = 1]$$

Definition 1 *A CUS scheme is said to be* sound *if for all polynomial time algorithms $\mathcal{A}$, the advantage $\texttt{Adv}^{sound}_{S,\mathcal{A}}$ is negligible in the security parameter k.*

Intuitively, the above soundness definition captures the requirement that a signer cannot "cheat" when interacting with a verifier in the confirm or disavow protocol i.e. he cannot convince a verifier that a signature is both valid and invalid. However, it does not guarantee that a cheating signer cannot produce a token tk_σ and a message/signature pair (m, σ) such that $\texttt{Verify}(par, pk, m, \sigma, tk_\sigma) = \texttt{accept}$, but (m, σ) can be disavowed. This requirement, which we will refer to as token soundness, often seems to be implicitly assumed and not formally defined in the used formalizations of the security for convertible undeniable signatures. Token soundness of a CUS scheme S is defined via the experiment $Exp_{S,\mathcal{A}}^{\texttt{tk-sound}}(1^k)$ shown in Figure 1. In a similar manner to above, we define the advantage of the algorithm $\mathcal{A}$ as

$$\texttt{Adv}_{S,\mathcal{A}}^{\texttt{tk-sound}} = \Pr[Exp_{S,\mathcal{A}}^{\texttt{tk-sound}}(1^k) = 1]$$

Definition 2 *A CUS scheme is said to have* token soundness *if for all polynomial time algorithms $\mathcal{A}$, the advantage $\texttt{Adv}_{S,\mathcal{A}}^{tk\text{-}sound}$ is negligible in the security parameter k.*

We note that the straightforward implementations of the recent schemes proposed by Phong, Kurosawa and Ogata [33] do not allow a verifier to check the correctness of a public key, and that this will essentially allow a malicious signer to break the token soundness of the scheme. More specifically, the schemes in [33] make use of NIZK proofs as tokens, and a signer will construct a token for a selective conversion of a signature by generating a NIZK proof of the validity of the signature. However, since the common reference string (CRS) used by the NIZK proofs is stored as part of the public signer key (i.e. the CRS will be generated by the signer), a malicious signer will, by the zero-knowledge property of the NIZK proofs, be able to generate a CRS which is indistinguishable from an honestly generated one, and which allows the signer to simulate the NIZK proofs. Hence, the malicious signer can break the token soundness of the schemes by simulating a NIZK proof for an invalid signature. We note that this vulnerability stems from the fact that a verifier cannot distinguish between an honestly generated CRS and a maliciously generated one, and is technically not a break of the token soundness as defined above since this definition assumes that the correctness of a public can be verified efficiently. This scenario can be avoid be letting the verifier generate the CRS, but this will tie a conversion to a single verifier and will not provide public verifiability. Alternatively, the CRS could be generate by a trusted third party. However, both of these options limit the practical applicability of the scheme.

4.2 Unforgeability

Strong unforgeability against a chosen message attack for a CUS scheme S is defined via the experiment $Exp_{S,\mathcal{A}}^{\texttt{suf-cma}}(1^k)$ shown in Figure 2. In the experiment, $\mathcal{A}$ has access to the oracle $\mathcal{O} = \{\mathcal{O}_{Sign}\}$ which is defined as follows

– $\mathcal{O}_{Sign}$: Given a message m, the oracle returns $\sigma \leftarrow \texttt{Sign}(par, sk, m)$.

It is required that $\mathcal{A}$ did not obtain σ^* by submitting m^* to $\mathcal{O}_{Sign}$. Note that since $\mathcal{A}$ is given the verification key vk, $\mathcal{A}$ can convert signatures and run the confirm and disavow protocols by himself, and there is no need to provide $\mathcal{A}$ with oracles for these tasks. The advantage of $\mathcal{A}$ is defined as

$$\texttt{Adv}_{S,\mathcal{A}}^{\texttt{suf-cma}} = \Pr[Exp_{S,\mathcal{A}}^{\texttt{suf-cma}}(1^k) = 1]$$

Definition 3 *A CUS scheme is said to be* strongly unforgeable *if for all polynomial time algorithms $\mathcal{A}$, the advantage $\texttt{Adv}_{S,\mathcal{A}}^{suf\text{-}cma}$ is negligible in the security parameter k.*

```
$\mathtt{Exp}^{\mathtt{suf\text{-}cma}}_{S,\mathcal{A}}(1^k)$
  $par \leftarrow \mathtt{Setup}(1^k)$
  $(pk, vk, sk) \leftarrow \mathtt{KeyGen}(par)$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk, vk)$
  if $(m^*, \sigma^*) \in \{(m, \sigma); \sigma \leftarrow \mathtt{Sign}(par, sk, m)\}$
    output 1
  else output 0
```

```
$\mathtt{Exp}^{\mathtt{inv\text{-}cma}}_{S,\mathcal{A}}(1^k)$
  $par \leftarrow \mathtt{Setup}(1^k)$
  $(pk, vk, sk) \leftarrow \mathtt{KeyGen}(par)$
  $(m^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk)$
  $b \leftarrow \{0, 1\}$
  if $b = 0$ set $\sigma^* \leftarrow \mathcal{S}$
  else set $\sigma^* \leftarrow \mathtt{Sign}(par, sk, m)$
  $b' \leftarrow \mathcal{A}^{\mathcal{O}}(st, \sigma^*)$
  if $b = b'$ output 1
  else output 0
```

Fig. 2. Unforgeability and invisibility experiments

While the above definition does not involve tokens, it will, in combination with token soundness, guarantee that an adversary without the knowledge of sk cannot produce (m, σ, tk_σ) such that $\mathtt{Verify}(par, pk, m, \sigma) = \mathtt{accept}$ without having obtained (m, σ) from the signer. This follows easily from the following observation. If the adversary does produce (m, σ, tk_σ), then if (m, σ) is a valid message/signature pair, the adversary has broken the above defined unforgeability property, whereas if (m, σ) is not a valid message/signature pair, the token soundness of the scheme has been broken, which should not be possible even for an adversary knowing sk.

We furthermore stress the importance of giving $\mathcal{A}$ access to vk in the above definition. This guarantees that if a signer delegates the verification operation by releasing vk to a semi-trusted entity, this entity will not be able to forge new signatures, but only verify existing ones.

4.3 Invisibility

Invisibility against a chosen message attack for a CUS scheme S is defined via the experiment $\mathtt{Exp}^{\mathtt{inv\text{-}cma}}_{S,\mathcal{A}}(1^k)$ shown in Figure 2. In the experiment, $\mathcal{S}$ denotes the signature space and is defined as $\mathcal{S} = \{\sigma : (pk, vk, sk) \leftarrow \mathtt{KeyGen}(par); m \leftarrow \mathcal{M}; \sigma \leftarrow \mathtt{Sign}(par, sk, m)\}$ where $\mathcal{M}$ is the message space given in par. Furthermore, $\mathcal{A}$ has access to the oracles $\mathcal{O} = \{\mathcal{O}_{Sign}, \mathcal{O}_{Conv}, \mathcal{O}_{Conf/Dis}\}$ which are defined as follows:

- $\mathcal{O}_{Sign}$: Defined as in the unforgeability experiment.
- $\mathcal{O}_{Conv}$: Given a message/signature pair (m, σ), this oracle returns $tk_\sigma \leftarrow \mathtt{Convert}(par, vk, m, \sigma)$ if $(m, \sigma) \in \{(m', \sigma') : \sigma' \leftarrow \mathtt{Sign}(par, sk, m)\}$. Otherwise the oracle returns $\perp$.
- $\mathcal{O}_{Conf/Dis}$: Given a message signature pair (m, σ), this oracle interacts with $\mathcal{A}$ by running $\mathtt{Confirm}(par, vk, m, \sigma)$ if $(m, \sigma) \in \{(m', \sigma') : \sigma' \leftarrow \mathtt{Sign}(par, sk, m)\}$. Otherwise, the oracle interacts with $\mathcal{A}$ by running $\mathtt{Disavow}(par, vk, m, \sigma)$.

It is required that $\mathcal{A}$ does not query (m^*, σ^*) to the convert or confirm/disavow oracles. We define the advantage of $\mathcal{A}$ in the experiment as

$$\mathtt{Adv}^{\mathtt{inv\text{-}cma}}_{S,\mathcal{A}} = |\Pr[Exp^{\mathtt{inv\text{-}cma}}_{S,\mathcal{A}}(1^k) = 1] - 1/2|$$

Definition 4 *A CUS scheme is said to be* invisible *if for all polynomial time algorithms $\mathcal{A}$ the advantage $\mathtt{Adv}^{inv\text{-}cma}_{S,\mathcal{A}}$ is negligible in the security parameter k.*

Note that the only requirement in the above definition is that $\mathcal{A}$ did not submit (m^*, σ^*) to the convert or confirm/disavow oracles. Hence, a deterministic scheme cannot satisfy the above definition since access to the signing oracle is not restricted (i.e. for a deterministic scheme, an adversary can simply submit m^* to $\mathcal{O}_{Sign}$ and compare the received signature with σ^*). Furthermore, $\mathcal{A}$ is allowed to submit (m^*, σ) where $\sigma \neq \sigma^*$, and (m, σ^*) where $m \neq m^*$ to the convert and confirm/disavow oracles. While a deterministic scheme should be able to satisfy a security definition where these type of queries are allowed, some security models (e.g. [38]) do not allow the former type of query, and thereby further weaken the obtain security. These issues might be a concern in a scenario where the entropy of the signed messages is small i.e. the security of a signature on a message which the signer has previously signed might not be guaranteed. However, with the above security notion, these concerns are eliminated.

Another aspect of the above security notion which we would like to highlight is the definition of $\mathcal{S}$. We note that anyone can sample $\mathcal{S}$ and that when using this definition of $\mathcal{S}$, invisibility implies anonymity i.e. the inability for an adversary to distinguish between signatures constructed by different users (see [17, 22] for a formal proof of this). Some schemes (e.g. [16, 23, 21]) use a more restricted definition limiting $\mathcal{S}$ to signatures from the signer i.e. $\mathcal{S} = \{\sigma : m \leftarrow \mathcal{M}; \sigma \leftarrow \texttt{Sign}(par, sk, m)\}$. This not only removes the guarantee of anonymity, but might also make it difficult for users other than the signer to sample $\mathcal{S}$. The latter can potentially have an impact on the non-transferability of the scheme, which we will define in the following section.

4.4 Non-transferability

The security notion non-transferability captures the property that a verifier who learns whether a given signature is valid or not by interacting with the signer in the confirm or disavow protocols, should not be able to prove this knowledge to a third party. More specifically, the verify should be able to "fake" any evidence of the validity of a signature obtain by interacting with the signer. When introducing convertible undeniable signatures, Boyar, Chaum, Damgård and Pedersen [7] referred to this property as *undeniability* and defined the property as a verifier's ability to produce fake signature/transcript pairs indistinguishable from real ones.

The above definition of invisibility guarantees that a valid signature from a signer cannot be distinguished from any other element in $\mathcal{S}$, and furthermore allows any user to sample $\mathcal{S}$. Hence, to obtain non-transferability, we will furthermore require that transcripts of the confirm and disavow protocols can be simulated. More specifically, we require that both $(\texttt{Confirm}, \texttt{V}_{con})$ and $(\texttt{Disavow}, \texttt{V}_{dis})$ are *computational zero-knowledge* (see [20] for a formal definition). Informally, this will guarantee that for any (possibly malicious) verifier, there exists an expected polynomial time simulator which can generate transcripts which are computational indistinguishable from transcripts obtained by interacting with a real prover. Hence, a verifier will be able to "fake" evidence of an interaction with a prover by running this simulator.

Definition 5 *A CUS scheme S is said to be* non-transferable *if S is invisible and the protocols* $(\texttt{Confirm}, V_{con})$ *and* $(\texttt{Disavow}, V_{dis})$ *are computational zero-knowledge.*

4.5 Non-impersonation

While soundness informally guarantees that a prover cannot "cheat", it does not prevent a third party from impersonating the prover. This was pointed out by Kurosawa and Henge

```
$Exp_{S,\mathcal{A}}^{\texttt{imp-cma}}(1^k)$
  $par \leftarrow \texttt{Setup}(1^k)$
  $(pk, vk, sk) \leftarrow \texttt{KeyGen}(par)$
  $(m^*, \sigma^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk)$
  if $(m^*, \sigma^*) \in \{(m, \sigma); \sigma \leftarrow \texttt{Sign}(par, sk, m)\}$
    $z \leftarrow_2 \{A(st) \leftrightarrow V_{con}\}(par, pk, m^*, \sigma^*)$
  else
    $z \leftarrow_2 \{A(st) \leftrightarrow V_{dis}\}(par, pk, m^*, \sigma^*)$
  if $z = \texttt{accept}$
    output 1
  else output 0
```

```
$Exp_{S,\mathcal{A}}^{\texttt{tk-imp-cma}}(1^k)$
  $par \leftarrow \texttt{Setup}(1^k)$
  $(pk, vk, sk) \leftarrow \texttt{KeyGen}(par)$
  $(m^*, \sigma^*, tk_\sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk)$
  if $\texttt{Verify}(par, pk, m^*, \sigma^*, tk_\sigma^*) = \texttt{accept}$
    output 1
  else output 0
```

Fig. 3. Non-impersonation experiments

[24], and Huang *et al.* [22] furthermore noticed that, for convertible schemes, this might be an issue for token generation as well.

Non-impersonation against a chosen message attack for a CUS scheme S is defined via the experiment $\texttt{Exp}_{S,\mathcal{A}}^{\texttt{imp-cma}}(1^k)$ shown in Figure 3. In the experiment, $\mathcal{A}$ has access to the oracles $\mathcal{O} = \{\mathcal{O}_{Sign}, \mathcal{O}_{Conv}, \mathcal{O}_{Conf/Dis}\}$ defined as the invisibility definition. It is required that $\mathcal{A}$ does not submit (m^*, σ^*) to the confirm/disavow oracle. We define the advantage of $\mathcal{A}$ in the experiment as

$$\texttt{Adv}_{S,\mathcal{A}}^{\texttt{imp-cma}} = \Pr[Exp_{S,\mathcal{A}}^{\texttt{imp-cma}}(1^k) = 1]$$

Definition 6 *A CUS scheme is said to be resistant to* impersonation attacks *if for all polynomial time algorithms $\mathcal{A}$, the advantage $\texttt{Adv}_{S,\mathcal{A}}^{\texttt{imp-cma}}$ is negligible in the security parameter k.*

Token non-impersonation against a chosen message attack for a CUS scheme S is defined via the experiment $\texttt{Exp}_{S,\mathcal{A}}^{\texttt{tk-imp-cma}}(1^k)$ shown in Figure 3. $\mathcal{A}$ has access to the oracles $\mathcal{O} = \{\mathcal{O}_{Sign}, \mathcal{O}_{Conv}, \mathcal{O}_{Conf/Dis}\}$ defined as in the above. It is required that $\mathcal{A}$ does not submit (m^*, σ^*) to the conversion oracle. The advantage of $\mathcal{A}$ in the experiment is defined as

$$\texttt{Adv}_{S,\mathcal{A}}^{\texttt{tk-imp-cma}} = \Pr[Exp_{S,\mathcal{A}}^{\texttt{tk-imp-cma}}(1^k) = 1]$$

Definition 7 *A CUS scheme is said to be resistant to* token impersonation attacks *if for all polynomial time algorithms $\mathcal{A}$, the advantage $\texttt{Adv}_{S,\mathcal{A}}^{\texttt{tk-imp-cma}}$ is negligible in the security parameter k.*

5 A Concrete Convertible Undeniable Signature Scheme

In this section we present a CUS scheme provable secure in the standard model. Our scheme has a similar structure to the basic designated confirmer signature scheme by Schuldt *et al.* [36], but we employ different proof systems and provide a token generation method. Furthermore, our scheme does not require verifiers to hold public/private key pairs and avoids the key registration requirement of [36]. In the description of the scheme we use the notation $ZKPK\{w : R\}$ to mean a zero-knowledge proof of knowledge of w such that the relation R holds. We present the implementation details of these proofs after describing the scheme.

– $\texttt{Setup}$: Compute $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi) \leftarrow \mathcal{G}(1^k)$, pick $g_2 \in \mathbb{G}_2$ and set $g_1 \leftarrow \psi(g_2)$. Furthermore, choose a collision resistant hash function family $\mathcal{H} = \{H_{\texttt{k}} : \{0,1\}^* \to \mathbb{Z}_p\}$ indexed by a key $\texttt{k} \in \mathcal{K}$. Return $par = (\mathbb{P}, g_1, g_2, \mathcal{H})$.

- `KeyGen` : Given par, pick $\alpha, x, y \leftarrow \mathbb{Z}_p$, $h \leftarrow \mathbb{G}_1$ and $w_2 \leftarrow \mathbb{G}_2$, and set $w_1 \leftarrow g_1^{\alpha}$, $v_1 \leftarrow g_2^{x^{-1}}$ and $v_2 \leftarrow g_2^{y^{-1}}$. Furthermore, pick $u_0, \ldots, u_n \leftarrow \mathbb{G}_2$[4], and define $F(m) = u_0 \prod_{i=1}^{n} u_i^{m_i}$ where m_i is the ith bit of m. Finally pick a hash key $\mathtt{k} \in \mathcal{K}$ and set $pk = (\mathtt{k}, w_1, w_2, v_1, v_2, h, u_0, \ldots, u_n)$, $vk = (x, y)$ and $sk = w_2^{\alpha}$. Return (pk, vk, sk).
- `Sign` : Given input (par, sk, m), where $sk = w_2^{\alpha}$, pick random $a, b, s \leftarrow \mathbb{Z}_p$, compute $t \leftarrow H_{\mathtt{k}}(\psi(v_1)^a || \psi(v_2)^b || m)$ and $M = g_1^s h^t$, and return $\sigma = (\psi(v_1)^a, \psi(v_2)^b, \psi(w_2^{\alpha} F(M)^{a+b}), s)$.
- `Convert` : Given (par, vk, m, σ) where $\sigma = (\sigma_1, \sigma_2, \sigma_3, s)$ and $vk = (x, y)$, check that $e(\sigma_3, g_2) = e(w_1, w_2) e(\sigma_1^x \sigma_2^y, F(M))$ where $M = g_1^s h^t$ and $t = H_{\mathtt{k}}(\sigma_1 || \sigma_2 || m)$, and return $\perp$ if this is not the case. Otherwise, return the token $tk_{\sigma} = (\sigma_1^x, \sigma_2^y)$.
- `Verify` : Given $(par, pk, m, \sigma, tk_{\sigma})$ where $pk = (\mathtt{k}, w_1, w_2, v_1, v_2, h, u_0, \ldots, u_n)$, $\sigma' = (\sigma_1, \sigma_2, \sigma_3, s)$ and $tk_{\sigma} = (tk_1, tk_2)$, return `accept` if $e(tk_1, v_1) = e(\sigma_1, g_2)$, $e(tk_2, v_2) = e(\sigma_2, g_2)$, and $e(\sigma_3, g_2) = e(w_1, w_2) e(tk_1 tk_2, F(M))$ where $M = g_1^s h^t$ and $t = H_{\mathtt{k}}(\sigma_1 || \sigma_2 || m)$.
- $(\mathtt{Confirm}, \mathtt{V}_{con})$: Given the common input (par, pk, m, σ), where $pk = (\mathtt{k}, w_1, w_2, v_1, v_2, h, u_0, \ldots, u_n)$ and $\sigma = (\sigma_1, \sigma_2, \sigma_3, s)$, and the additional private input $vk = (x, y)$ to the `Confirm` algorithm, $(\mathtt{Confirm}, V_{con})$ is executed as

  $$ZKPK\{(x, y) : v_1^x = g_2 \wedge v_2^y = g_2 \wedge e(\sigma_1, F(M))^x e(\sigma_2, F(M))^y = e(\sigma_3, g_2)/e(w_1, w_2)\}$$

  where $M = g_1^s h^t$ and $t = H_{\mathtt{k}}(\sigma_1 || \sigma_2 || m)$.
- $(\mathtt{Disavow}, \mathtt{V}_{dis})$: Given same common input as in $(\mathtt{Confirm}, V_{con})$ and private input $vk = (x, y)$ to the `Disavow` algorithm, $(\mathtt{Disavow}, V_{dis})$ is executed as

  $$ZKPK\{(x, y) : v_1^x = g_2 \wedge v_2^y = g_2 \wedge e(\sigma_1, F(M))^x e(\sigma_2, F(M))^y \neq e(\sigma_3, g_2)/e(w_1, w_2)\}$$

  where $M = g_1^s h^t$ and $t = H_{\mathtt{k}}(\sigma_1 || \sigma_2 || m)$.

Implementation of ZKPK. To implement the zero-knowledge proofs of knowledge for the above scheme, we employ a transformation proposed by Cramer, Damgård and MacKenzie [14] which converts a sigma protocol into a perfect zero-knowledge proof of knowledge. As a tool, the transform makes use of the well known technique by Cramer, Damgård and Shoenmakers [15] for constructing a witness indistinguishable "OR" proof from sigma protocols i.e. a sigma protocol Σ for relation R and a sigma protocol Σ' for relation R' is combined into a sigma protocol, denoted $\Sigma \vee \Sigma'$, which for common input (x, x') and a witness w, proves that either $(x, w) \in R$ or $(x', w) \in R'$. Given that Σ and Σ' both have challenge space $\mathbb{Z}_p$, and if we, without loss of generality, assume that the prover knows a witness w such that $(x, w) \in R$, $\Sigma \vee \Sigma'$ is implemented as follows.

1. The prover honestly computes the first message a of Σ, picks random c' and runs $(a', c', z') \leftarrow \mathtt{Sim}_{\Sigma'}(x', c')$. Then (a, a') is sent to the verifier.
2. The verifier sends a randomly chosen $\bar{c}$ to prover.
3. The prover computes $c \leftarrow \bar{c} - c' \mod p$, respond honestly to c with the message z following Σ, and sends (c, c', z, z') to the verifier who checks that $\bar{c} = c + c' \mod p$ and that (a, c, z) and (a', c', z') are accepting transcripts of Σ and Σ'.

In the above protocol, the verifier is unable to determine which relation the prover holds a witness for. We refer the reader to [15] for a detailed analysis of the protocol.

The transformation for obtaining a zero-knowledge proof of knowledge works as follows. Let Σ be a sigma protocol for a relation R with messages (a, c, z). Consider the following commitment scheme induced by Σ and a fixed common input x:

[4] We assume that the description of group elements in $\mathbb{G}_1$ is less than n bits.

– To commit to a value c in the challenge space of Σ, simulate a transcript $(a, c, z) \leftarrow \mathtt{Sim}_{\Sigma}(x, c)$, and return a as a commitment on c.
– To open a commitment a, reveal the corresponding (c, z).
– A verifier verifies the opening by confirming that (a, c, z) is an accepting transcript of Σ.

Furthermore, for this commitment scheme, assume there is a sigma protocol Σ' for proving knowledge of a committed value i.e. a sigma protocol for the relation R' with common input $x' = (x, a)$ and witness $w' = (c, z)$ (this is the only needed assumption). Then we can obtain a perfect zero-knowledge proof for $(x, w) \in R$, where x is the common input and w is the witness held by the prover, as follows:

1. Using the commitment scheme induced by Σ and x, the verifier commits to a random value in the challenge space of Σ, and then proves knowledge of this value to the prover using Σ'. (If the verifier does not provide an accepting proof, the prover will abort.) Let *com* denote the commitment sent by the verifier.
2. The prover then proves to the verifier that he knows either the witness w or an opening of *com* using the sigma protocol $\Sigma \vee \Sigma'$.

Although the above protocol is a six move protocol, it can easily be reduced to a four move protocol by combining the second and third moves of the sigma protocol in step 1 with the first and second moves of the sigma protocol in step 2.

In [14], Cramer, Damgård and MacKenzie show that the protocol resulting from the above transformation is a perfect zero-knowledge proof of knowledge with knowledge error at most 2^{-t} assuming Σ uses t-bit challenges. We refer the reader to [14] for the details.

To obtain a zero-knowledge proof for $(\mathtt{Confirm}, \mathtt{V}_{con})$ with common input (par, pk, σ, m), we construct the following sigma protocol (as mentioned in Section 2, the construction of this type of sigma protocol is relatively straightforward):

$$\Sigma_{Std}\{(x, y) : v_1^x = g_2 \wedge v_2^y = g_2 \wedge e(\sigma_1, F(M))^x e(\sigma_2, F(M))^y = e(\sigma_3, g_2)/e(w_1, w_2)\}$$

where $M = g_1^s h^t$ and $t = H_{\mathtt{k}}(\sigma_1||\sigma_2||m)$. A commitment to a value $c \in \mathbb{Z}_p$ constructed using Σ_{Std} and (par, pk, σ, m) will be of the form $(a_1, a_2, a_3) = (v_1^{z_1} g_2^{-c}, v_2^{z_2} g_2^{-c}, e_1^{z_1} e_2^{z_2} e_3^{-c})$ where $e_1 = e(\sigma_1, F(M))$, $e_2 = e(\sigma_2, F(M))$, $e_3 = e(\sigma_3, g_2)/e(w_1, w_2)$, and $z_1, z_2 \leftarrow \mathbb{Z}_p$ are random values. The corresponding opening is (c, z_1, z_2), and a verifier accept the opening if $(a_1, a_2, a_3, c, z_1, z_2)$ is a valid transcript of Σ_{Std}. The following sigma protocol proves knowledge of an opening.

$$\Sigma'_{Std}\{(c, z_1, z_2) : v_1^{z_1} g_2^{-c} = a_1 \wedge v_2^{z_2} g_2^{-c} = a_2 \wedge e_1^{z_1} e_2^{z_2} e_3^{-c} = a_3\}$$

Hence, by applying the above transformation to Σ_{Std} and Σ'_{Std}, a zero-knowledge proof for $(\mathtt{Confirm}, \mathtt{V}_{con})$ is obtained.

To obtain a zero-knowledge proof for $(\mathtt{Disavow}, V_{dis})$ with common input (par, pk, m, σ), we apply the same strategy as above. Firstly, we construct a sigma protocol $\Sigma\{(x, y) : v_1^x = g_2 \wedge v_2^y = g_2 \wedge e_1^x e_2^y \neq e_3\}$ by adapting the technique used in the proof of inequality of discrete logarithms by Camenish and Shoup [9]. More specifically, a prover first chooses $r \leftarrow \mathbb{Z}_p$, computes $C \leftarrow (e_1^x e_2^y / e_3)^r$, and then interacts with the verifier in the protocol

$$\overline{\Sigma}_{Std}\{(\alpha, \beta, r) : v_1^{\alpha} g_2^{-r} = 1 \wedge v_2^{\beta} g_2^{-r} = 1 \wedge e_1^{\alpha} e_2^{\beta} e_3^{-r} = C\}$$

where $\alpha = xr$ and $\beta = yr$. A verifier will only accept a proof if $C \neq 1$. A commitment to a value $c \in \mathbb{Z}_p$ constructed using $\overline{\Sigma}_{Std}$ and (par, pk, σ, m) is of the form $(a_1, a_2, a_3, C) = (v_1^{z_\alpha} g_2^{-z_r}, v_2^{z_\beta} g_2^{-z_r}, e_1^{z_\alpha} e_2^{z_\beta} e_3^{-z_r} C^{-c}, C)$ where $z_\alpha, z_\beta, z_r \leftarrow \mathbb{Z}_p$ and $C \leftarrow \mathbb{G}_T$ are random values. To prove knowledge of the opening $(c, z_\alpha, z_\beta, z_r)$, the following sigma protocol can be used

$$\overline{\Sigma}'_{Std}\{(c, z_\alpha, z_\beta, z_r) : v_1^{z_\alpha} g_2^{-z_r} = a_1 \wedge v_2^{z_\beta} g_2^{-z_r} = a_2 \wedge e_1^{z_\alpha} e_2^{z_\beta} e_3^{-z_r} C^{-c} = a_3\}$$

Hence, by applying the above transformation to $\overline{\Sigma}_{Std}$ and $\overline{\Sigma}'_{Std}$ we obtain a zero-knowledge proof for $(\texttt{Disavow}, \texttt{V}_{dis})$.

5.1 Security

The soundness of the above scheme and the zero-knowledge property of the confirm and disavow protocols follows directly from the soundness and the zero-knowledge properties of the proofs obtained from the transformation by Cramer, Damgård and MacKenzie [14]. We refer the reader to [14] for proofs of these properties.

The token soundness of the scheme is implied by the properties of the bilinear map in combination with the proof of knowledge property of the disavow protocol (which is achieved without requiring any intractability assumptions [14]).

Theorem 8 *If the disavow protocol is a zero-knowledge proof of knowledge, then the above CUS scheme has token soundness.*

The proof is given in Appendix B.

The unforgeability of the above CUS scheme is based on the unforgeability of the signature scheme by Waters [37] which we recall in Appendix A. We note that Waters' signature scheme is (weakly) unforgeable assuming the computational co-Diffie-Hellman problem is hard [37]. The proof of the first theorem below, which is given in Appendix C, is based on the ideas by Schuldt and Matsuura [36] which in turn is based on ideas from Boneh, Shen and Waters [6]. The proof of the second theorem is likewise based on ideas from [36] and is given in Appendix D.

Theorem 9 *Assume that the hash function family $\mathcal{H}$ is collision resistant, the discrete logarithm problem is hard in $\mathbb{G}_2$, and that Waters signature scheme is (weakly) unforgeable. Then the above CUS scheme is strongly unforgeable.*

Theorem 10 *Assume the above CUS scheme is strongly unforgeable and that the decision linear problem is hard in $\mathbb{G}_2$. Then the above CUS scheme is invisible.*

Lastly, the following theorems show that the above CUS scheme is resistant to impersonation and token impersonation attacks. The strategy of the proof of the first theorem is to make use of the proof of knowledge property of the used proof systems to extract the verification key which, in the simulation, will contain an unknown discrete logarithm. Otherwise, the simulation is very similar to that of Theorem 10, and can easily be derived from this. The strategy of the proof of the second theorem is to return a signature in one of the adversary's signature queries such that a conversion will reveal the solution to a computational ψ-Diffie-

Hellman problem[5]. The strong unforgeability of the scheme will ensure that the signature converted by the adversary in a token impersonation attack was constructed by the simulator, and the remaining part of the simulation is similar to that of Theorem 10. We omit the details of the proofs here.

Theorem 11 *Assume the above CUS scheme is strongly unforgeable and that the the discrete logarithm problem is hard in* $\mathbb{G}_2$*. Then the above CUS scheme is resistant to impersonation attacks.*

Theorem 12 *Assume the above CUS scheme is strongly unforgeable and that the computational* ψ*-Diffie-Hellman problem is hard in* $(\mathbb{G}_1, \mathbb{G}_2)$*. Then the above CUS scheme is resistant to token impersonation attacks.*

References

1. Laila El Aimani. Toward a generic construction of universally convertible undeniable signatures from pairing-based signatures. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 2008.
2. Laila El Aimani. Toward a generic construction of convertible undeniable signatures from pairing-based signatures. Cryptology ePrint Archive, Report 2009/362, 2009. `http://eprint.iacr.org/`.
3. Laila El Aimani and Damien Vergnaud. Gradually convertible undeniable signatures. In Jonathan Katz and Moti Yung, editors, *ACNS*, volume 4521 of *Lecture Notes in Computer Science*, pages 478–496. Springer, 2007.
4. Dan Boneh, editor. *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*. Springer, 2003.
5. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.
6. Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational diffie-hellman. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2006.
7. Joan Boyar, David Chaum, Ivan Damgård, and Torben P. Pedersen. Convertible undeniable signatures. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 189–205. Springer, 1990.
8. Gilles Brassard, editor. *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*. Springer, 1990.
9. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Boneh [4], pages 126–144.
10. Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report 260, Institute for Theoretical Computer Science, ETH Zurich, March 1997.
11. David Chaum. Zero-knowledge undeniable signatures. In *EUROCRYPT*, pages 458–464, 1990.
12. David Chaum and Hans Van Antwerpen. Undeniable signatures. In Brassard [8], pages 212–216.
13. Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.
14. Ronald Cramer, Ivan Damgård, and Phillip MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *Public Key Cryptography — PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 354–372. Springer-Verlag, January 2000.

[5] More specifically, a simulator given $\mathbb{P}$ and a ψ-Diffie-Hellman instance $g, g^a, h' \in \mathbb{G}_2$ will set $g_2 \leftarrow g^a$, $v_1 \leftarrow g$, $u_i \leftarrow g^{d_i}$ with random $d_i \leftarrow \mathbb{Z}_p$ for $0 \leq i \leq n$, and generate the remaining elements of par and pk as in an ordinary scheme. Then, for a message m, the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, s)$ constructed by picking $r, b, s \leftarrow \mathbb{Z}_p$, setting $\sigma_1 \leftarrow \psi(h')^r$, $\sigma_2 \leftarrow \psi(v_2)^b$, $t \leftarrow H_{\mathtt{k}}(\sigma_1||\sigma_2||m)$, $M \leftarrow g_1^s h^t$ (where h is from pk) and $\sigma_3 \leftarrow \psi(w_2^\alpha (h')^{r(d_0+\sum_{i=1}^n d_i M_i)} F(M)^b)$, will have a conversion token of the form $tk_\sigma = (\psi(h')^{ra}, g_1^b)$ for a known value r.

15. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.
16. Ivan Damgård and Torben P. Pedersen. New convertible undeniable signature schemes. In *EUROCRYPT*, pages 372–386, 1996.
17. Steven D. Galbraith and Wenbo Mao. Invisibility and anonymity of undeniable and confirmer signatures. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 2003.
18. Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Rsa-based undeniable signatures. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 132–149. Springer, 1997.
19. Rosario Gennaro, Tal Rabin, and Hugo Krawczyk. Rsa-based undeniable signatures. *J. Cryptology*, 20(3):394, 2007.
20. O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.
21. Xinyi Huang, Yi Mu, Willy Susilo, and Wei Wu. A generic construction for universally-convertible undeniable signatures. In Feng Bao, San Ling, Tatsuaki Okamoto, Huaxiong Wang, and Chaoping Xing, editors, *CANS*, volume 4856 of *Lecture Notes in Computer Science*, pages 15–33. Springer, 2007.
22. Xinyi Huang, Yi Mu, Willy Susilo, and Wei Wu. Provably secure pairing-based convertible undeniable signature with short signature length. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing*, volume 4575 of *Lecture Notes in Computer Science*, pages 367–391. Springer, 2007.
23. Kaoru Kurosawa and Jun Furukawa. Universally composable undeniable signature. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 524–535. Springer, 2008.
24. Kaoru Kurosawa and Swee-Huay Heng. 3-move undeniable signature scheme. In Cramer [13], pages 181–197.
25. Kaoru Kurosawa and Tsuyoshi Takagi. New approach for selectively convertible undeniable signature schemes. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 428–443. Springer, 2006.
26. Fabien Laguillaumie and Damien Vergnaud. Short undeniable signatures without random oracles: The missing link. In Subhamoy Maitra, C. E. Veni Madhavan, and Ramarathnam Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 283–296. Springer, 2005.
27. Fabien Laguillaumie and Damien Vergnaud. Time-selective convertible undeniable signatures. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 154–171. Springer, 2005.
28. Markus Michels, Holger Petersen, and Patrick Horster. Breaking and repairing a convertible undeniable signature scheme. In *ACM Conference on Computer and Communications Security*, pages 148–152, 1996.
29. Markus Michels and Markus Stadler. Efficient convertible undeniable signature schemes (extended abstract). pages 231–244. Springer-Verlag, 1997.
30. Takeru Miyazaki. An improved scheme of the gennaro-krawczyk-rabin undeniable signature system based on rsa. In Dongho Won, editor, *ICISC*, volume 2015 of *Lecture Notes in Computer Science*, pages 135–149. Springer, 2000.
31. Jean Monnerat and Serge Vaudenay. Generic homomorphic undeniable signatures. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 354–371. Springer, 2004.
32. Jean Monnerat and Serge Vaudenay. Short 2-move undeniable signatures. In Phong Q. Nguyen, editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 2006.
33. Le Trieu Phong, Kaoru Kurosawa, and Wakaha Ogata. New dlog-based convertible undeniable signature schemes in the standard model. Cryptology ePrint Archive, Report 2009/394, 2009. `http://eprint.iacr.org/`.
34. Le Trieu Phong, Kaoru Kurosawa, and Wakaha Ogata. New rsa-based (selectively) convertible undeniable signature schemes. In Bart Preneel, editor, *AFRICACRYPT*, volume 5580 of *Lecture Notes in Computer Science*, pages 116–134. Springer, 2009.
35. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Brassard [8], pages 239–252.
36. Jacob C. N. Schuldt and Kanta Matsuura. On-line non-transferable signatures revisited. Cryptology ePrint Archive, Report 2009/406, 2009. `http://eprint.iacr.org/`.
37. Brent Waters. Efficient identity-based encryption without random oracles. In Cramer [13], pages 114–127.
38. Tsz Hon Yuen, Man Ho Au, Joseph K. Liu, and Willy Susilo. (convertible) undeniable signatures without random oracles. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS*, volume 4861 of *Lecture Notes in Computer Science*, pages 83–97. Springer, 2007.

A Waters' Signatures

Below we recall the signature scheme by Waters [37]. Note that we make use of an asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ whereas the original scheme in [37] was defined using a symmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$.

- `Setup`: Compute $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi) \leftarrow \mathcal{G}(1^k)$, pick $g_2 \in \mathbb{G}_2$ and set $g_1 \leftarrow \psi(g_2)$. Return $par = (\mathbb{P}, g_1, g_2)$.
- `KeyGen` : Given par, pick $\alpha \leftarrow \mathbb{Z}_p$ and $w_2 \leftarrow \mathbb{G}_2$, and set $w_1 \leftarrow g_1^\alpha$. Furthermore, pick $u_0, \ldots, u_n \leftarrow \mathbb{G}_2$, and define $F(m) = u_0 \prod_{i=1}^n u_i^{m_i}$ where m_i is the ith bit of m. Finally set the public key to $pk = (w_1, w_2, u_0, \ldots, u_n)$ and the private key to $sk = w_2^\alpha$. Return (pk, sk).
- `Sign` : Given input (par, sk, m), where $sk = w_2^\alpha$ pick $r \leftarrow \mathbb{Z}_p$, compute $\sigma_1 \leftarrow g_1^r$ and $\sigma_2 \leftarrow \psi(w_2^\alpha F(m)^r)$, and return the signature $\sigma = (\sigma_1, \sigma_2)$.
- `Verify` : Given par, a public key $pk = (w_1, w_2, u_0, \ldots, u_n)$, a message m and a signature $\sigma = (\sigma_1, \sigma_2)$, return `accept` if $e(\sigma_2, g_2) = e(w_1, w_2)e(\sigma_1, F(m))$.

It follows from the proof of security given in [37], that the above signature scheme is unforgeable against a chosen message attack assuming the computational co-Diffie-Hellman problem is hard in $(\mathbb{G}_1, \mathbb{G}_2)$.

B Proof of Theorem 8

Proof. Firstly, we show that if a signature passes the token verification, it must be a valid signature. Consider a public key $pk = (\mathtt{k}, w_1, w_2, v_1, v_2, h, u_0, \ldots, u_n)$, a purported signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, s)$ on a message m, and a token $tk_\sigma = (tk_1, tk_2)$. There must exist unique values $a, b \in \mathbb{Z}_p$ such that $\sigma_1 = \psi(v_1)^a$ and $\sigma_2 = \psi(v_2)^b$. Then, if the verification equations $e(tk_1, v_1) = e(\sigma_1, g_2)$ and $e(tk_2, v_2) = e(\sigma_2, g_2)$ hold, we must have that $tk_1 = g_1^a$ and $tk_2 = g_1^b$. Hence, we have $tk_1 tk_2 = g_1^{a+b}$. Furthermore, if $e(\sigma_3, g_2) = e(w_1, w_2)e(g_1^{a+b}, F(M))$, where $M = g_1^s h^t$ and $t = H_\mathtt{k}(\sigma_1||\sigma_2||m)$, also holds, we have that $\sigma_3 = \psi(w_2^\alpha F(M)^{a+b})$, where $\alpha = \log_{g_1} w_1$. Hence, if the output of $\mathtt{Verify}(par, pk, m, \sigma, tk_\sigma)$ is `accept`, σ must be a valid signature on m under the public key pk.

Now assume that the adversary completes the disavow protocol for (pk, m, σ) with non-negligible probability. By the proof of knowledge property of the protocol, there exists a knowledge extractor which is able to extract a witness (α, β, r) with non-negligible probability such that $v_1^\alpha = g_2^r$, $v_2^\beta = g_2^r$ and $e_1^\alpha e_2^\beta / e_3^r = C$ for some $C \neq 1$, where $e_1 = e(\sigma_1, F(M))$, $e_2 = e(\sigma_2, F(M))$, and $e_3 = e(\sigma_3, g_2)/e(w_1, w_2)$. Letting $x = \log_{v_1} g_2$ and $y = \log_{v_2} g_2$, this implies that $\alpha = xr$ and $\beta = yr$. Hence, we must have that $(e_1^x e_2^y / e_3)^r \neq 1$ which implies that $e_1^x e_2^y \neq e_3$. However, this contradicts that σ is a valid signature on m under pk.

C Proof of Theorem 9

Proof. Assume that a successful adversary $\mathcal{A}$ that breaks the unforgeability of the CUS scheme exists. Let (m^*, σ^*) denote the forgery output by $\mathcal{A}$ where $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, s^*)$, and let (m_i, σ_i) denote the ith sign query and response where $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}, s_i)$. Furthermore, we let $M^* = g_1^{s^*} h^{t^*}$, $t^* = H_\mathtt{k}(\sigma_1^*||\sigma_2^*||m^*)$, $M_i = g_1^{s_i} h^{t_i}$, and $t_i = H(\sigma_{i,1}||\sigma_{i,2}||m_i)$, where h is part of the public key pk. Finally, let q be the total number of sign queries made the adversary. We then define three different types of forgeries:

1. A forgery where $M^* = M_i$ and $t^* = t_i$ for some $i \in \{1, \ldots, q\}$.
2. A forgery where $M^* = M_i$ and $t^* \neq t_i$ for some $i \in \{1, \ldots, q\}$.
3. A forgery where $M^* \neq M_i$ for all $i \in \{1, \ldots, q\}$.

If $\mathcal{A}$ is successful, he must produce a forgery belonging to one of the above categories. For each category, we define algorithms $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ that breaks the collision resistance of $H_{\mathtt{k}}$, solves the discrete logarithm problem in $\mathbb{G}_2$, and breaks the weak unforgeability of the Waters signature scheme, respectively.

Category 1. $\mathcal{B}_1$'s goal for this category forgery is, given a description of a hash function family $\mathcal{H}$ and a random hash key $\mathtt{k} \in \mathcal{K}$, to produce messages $x_1 \neq x_2$ such that $H_{\mathtt{k}}(x_1) = H_{\mathtt{k}}(x_2)$. We construct $\mathcal{B}_1$ as follows: Firstly, $\mathcal{B}_1$ runs $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi) \leftarrow \mathcal{G}(1^k)$, picks $g_2 \leftarrow \mathbb{G}_2$ and sets $g_1 \leftarrow \psi(g_2)$ and $par \leftarrow (\mathbb{P}, g_1, g_2, \mathcal{H})$. Then $\mathcal{B}_1$ runs $(pk, vk, sk) \leftarrow \mathtt{KeyGen}(par)$ but uses the received hash key $\mathtt{k}$ in pk instead of picking a random key. Then $\mathcal{B}_1$ runs $\mathcal{A}$ with input (par, pk, vk).

While running, $\mathcal{A}$ can ask sign queries m_i which $\mathcal{B}_1$ responds to by returning $\sigma_i \leftarrow \mathtt{Sign}(par, sk, m_i)$. Eventually, $\mathcal{A}$ outputs a forgery (m^*, σ^*) where σ^* is of the form $(\sigma_1^*, \sigma_2^*, \sigma_3^*, s^*)$. $\mathcal{B}_1$ outputs messages $x_1 = \sigma_1^* || \sigma_2^* || m^*$ and $x_2 = \sigma_{i,1} || \sigma_{i,2} || m_i$ where i is the index for which $M^* = M_i$ and $t^* = t_i$.

$\mathcal{B}_1$ succeeds if $x_1 \neq x_2$. Assume towards a contradiction that $x_1 = x_2$ i.e. $\sigma_1^* || \sigma_2^* || m^* = \sigma_{i,1} || \sigma_{i,2} || m_k$. Since $\mathcal{A}$'s forgery is a type 1 forgery, we have that $M_i = g^{s_i} h^{t_i} = g^{s^*} h^{t^*} = M^*$ and $t_i = t^*$. This implies that $s_i = s^*$. Furthermore, if σ^* is a valid signature, it must be possible to write σ_3^* as $\psi(w_2^\alpha F(M^*)^{a+b})$ where $a = \log_{v_1} \sigma_1^*$ and $b = \log_{v_2} \sigma_2^*$. However, since $\sigma_1^* = \sigma_{i,1}$, $\sigma_2^* = \sigma_{i,2}$ and $M^* = M_i$, we must have $\sigma_{i,3} = \psi(w_2^\alpha F(M_i)^{a+b}) = \sigma_3^*$. Hence, we have $(\sigma_1^*, \sigma_2^*, \sigma_3^*, s^*) = (\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}, s_i)$ which contradicts $\mathcal{A}$ outputting a valid forgery.

Category 2. $\mathcal{B}_2$'s goal in this category is, given $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi)$ and elements $g_2, h_2 \in \mathbb{G}_2$, to compute $x = \log_{g_2} h_2$. We construct $\mathcal{B}_2$ as follows: Firstly, $\mathcal{B}_2$ picks a hash function family $\mathcal{H}$, sets $g_1 \leftarrow \psi(g_2)$ and $par \leftarrow (\mathbb{P}, g_1, g_2, \mathcal{H})$, and runs $(pk, vk, sk) \leftarrow \mathtt{KeyGen}(par)$ but uses the element $h \leftarrow \psi(h_2)$ in pk instead of picking a random element in $\mathbb{G}_1$. Then $\mathcal{B}_2$ runs $\mathcal{A}$ with input (par, pk, vk).

While running, $\mathcal{A}$ can ask sign queries which $\mathcal{B}_2$ responds to by returning $\sigma_i \leftarrow \mathtt{Sign}(par, sk, m_i)$. Eventually, $\mathcal{A}$ outputs a forgery (m^*, σ^*). Since $\mathcal{A}$'s forgery is assumed to be of type 2, there must be a i such that $M_i = g_1^{s_i} h^{t_i} = g_1^{s^*} h^{t^*} = M^*$ but $t_i \neq t^*$. Hence, $\mathcal{B}_2$ can compute $x = \log_{g_1} h = \log_{g_2} h_2 = (s_i - s^*)/(t^* - t_i)$.

Category 3. $\mathcal{B}_3$'s goal in this category is to produce a forgery of the signature scheme by Waters. $\mathcal{B}_3$ interacts with a (weak) unforgeability challenger $\mathcal{C}$ which provides $\mathcal{B}_3$ with a signing oracle. Initially, $\mathcal{B}_3$ is given parameters $par = (\mathbb{P}, g_1, g_2, \mathcal{H})$ and a public key $pk' = (w_1, w_2, u_0, \ldots, u_n)$. Firstly, $\mathcal{B}_3$ creates a public key pk by picking a hash key $\mathtt{k} \in \mathcal{K}$, picking values $c, x, y \leftarrow \mathbb{Z}_p$, setting $h \leftarrow g_1^c$, $v_1 \leftarrow g_2^{-x}$ and $v_2 \leftarrow g_2^{-y}$, and finally setting $pk = (\mathtt{k}, w_1, w_2, v_1, v_2, h, u_0, \ldots, u_n)$. Then $\mathcal{B}_3$ sets $vk \leftarrow (x, y)$ and runs $\mathcal{A}$ with input (par, pk, vk).

While running, $\mathcal{A}$ can ask sign queries, which $\mathcal{B}_3$ responds to as follows:

– *Sign queries*: Given a message m, $\mathcal{B}_3$ picks random $z \leftarrow \mathbb{Z}_p$, sets $M \leftarrow g_1^z$ and queries M to its signing oracle to obtain a Waters signature $\sigma' = (g_1^r, \psi(w_2^\alpha F(M)^r))$ on M. Then $\mathcal{B}_3$ picks random $r' \leftarrow \mathbb{Z}_p$, and computes $\sigma_1 = \psi(v_1)^{r+r'} \leftarrow (g_1^r g_1^{r'})^{x^{-1}}$ and $\sigma_2 = \psi(v_2)^{r-r'} \leftarrow (g_1^r / g_1^{r'})^{y^{-1}}$. Note that since both r and r' will be uniformly distributed in $\mathbb{Z}_p$, so will

$r + r'$ and $r - r'$ (like the values a and b in an ordinary signature). Lastly, $\mathcal{B}_3$ computes $s \leftarrow z - cH_{\mathtt{k}}(\sigma_1||\sigma_2||m)$, sets $\sigma = (\sigma_1, \sigma_2, \psi(w_2^\alpha F(M)^r), s)$, and returns σ to $\mathcal{A}$. Note that s is uniformly distributed in $\mathbb{Z}_p$ and that $M = g^s h^{H_{\mathtt{k}}(\sigma_1||\sigma_2||m)}$. Hence, σ is a valid signature on m.

Eventually, $\mathcal{A}$ outputs a forgery (m^*, σ^*) where $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, s^*)$. $\mathcal{B}_3$ then computes $\sigma_1' \leftarrow (\sigma_1^*)^x (\sigma_2^*)^y$ and outputs the message M^* and $\sigma' \leftarrow (\sigma_1', \sigma_3^*)$. Note that if $\mathcal{A}$'s forgery is valid, it follows that $e(\sigma_3^*, g_2) = e(w_1, w_2)e(\sigma_1', F(M^*))$, and since $\mathcal{A}$'s forgery is assumed to be a type 3 forgery, we have that $M^* \neq M_i$ for all i. Hence, the validity of $\mathcal{B}_3$'s forgery follows directly from the validity of $\mathcal{A}$'s forgery, and $\mathcal{B}_3$ successfully attacks the weak unforgeability of the Waters signature scheme whenever $\mathcal{A}$ successfully attacks the CUS scheme. □

D Proof of Theorem 10

Proof. We assume that an adversary $\mathcal{A}$ breaking the invisibility of the CUS scheme exists. Let `forge` be the event that $\mathcal{A}$ submits a convert or confirm/disavow query (m, σ) where σ is a valid signature on m which was not obtained through a sign query m. In the following we will construct algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ which will break the strong unforgeability of the scheme and the linear assumption in the events `forge` and $\neg$`forge`, respectively.

Firstly assume that the event `forge` happens. $\mathcal{B}_1$ runs an unforgeability experiment, receives the input (par, pk, vk), and forwards (par, pk) as input to $\mathcal{A}$. While running, $\mathcal{A}$ can ask sign, convert and confirm/disavow queries. $\mathcal{B}_1$ responds to these queries as follows. If $\mathcal{A}$ makes a sign query, $\mathcal{B}_1$ forwards this queries to his own signing oracle, and returns the obtained signature to $\mathcal{A}$. If $\mathcal{A}$ makes a convert or confirm/disavow query (m, σ), $\mathcal{B}_1$ first checks if σ was returned as a response to a sign query on m. If this is not the case, $\mathcal{B}_1$ checks if (m, σ) is valid (using vk), and if so, returns (m, σ) as a forgery and halts. Otherwise, $\mathcal{B}_1$ either returns $tk_\sigma \leftarrow$ `Convert`(par, vk, m, σ) or $\perp$, or interacts with $\mathcal{A}$ running `Confirm` or `Disavow`, depending on the query type and the validity of (m, σ).

At some point, $\mathcal{A}$ outputs a challenge message m^*. As in the invisibility experiment, $\mathcal{B}_1$ flips a random coin $b \leftarrow \{0, 1\}$ and returns a random $\sigma^* \leftarrow \mathcal{S}$ if $b = 0$. Otherwise, $\mathcal{B}_1$ returns σ^* obtained by submitting m^* to his own signing oracle. After receiving σ^*, $\mathcal{A}$ can ask additional sign, convert and confirm/disavow queries which $\mathcal{B}_1$ answers as above. If `forge` happens, it is clear that $\mathcal{B}_1$ succeeds in winning in the unforgeability experiment.

Now assume that `forge` does not happen. $\mathcal{B}_2$ will attempt to solve the decisional linear assumption i.e. $\mathcal{B}_2$ receives $\mathbb{P} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi)$ and elements $u, v, u^x, v^y, h, h^z \in \mathbb{G}_2$. $\mathcal{B}_2$'s goal is to decide if $z = x + y$. Firstly, $\mathcal{B}_2$ picks a hash family $\mathcal{H} = \{H_{\mathtt{k}} : \{0,1\}^* \to \mathbb{Z}_p\}$ and an element $g_2 \leftarrow \mathbb{G}_2$, and sets $g_1 \leftarrow \psi(g_2)$ and $par \leftarrow (\mathbb{P}, g_1, g_2, \mathcal{H})$. $\mathcal{B}_2$ then generates a public key by choosing $\alpha \leftarrow \mathbb{Z}_p$ and $w_2 \leftarrow \mathbb{G}_2$, and setting $w_1 \leftarrow g_1^\alpha$, $v_1 \leftarrow u$, $v_2 \leftarrow v$ and $h_1 \leftarrow \psi(h)$. Furthermore, $\mathcal{B}_2$ picks a hash key $\mathtt{k} \in \mathcal{K}$ and $d_0, \ldots, d_n \leftarrow \mathbb{Z}_p$, and sets $u_i \leftarrow h^{d_i}$ for $1 \leq i \leq n$, $pk \leftarrow (\mathtt{k}, w_1, w_2, v_1, v_2, h_1, u_0, \ldots, u_n)$ and $sk = w_2^\alpha$. Lastly, $\mathcal{B}_2$ runs $\mathcal{A}$ with input (par, pk).

While running, $\mathcal{A}$ can ask sign, convert and confirm/disavow queries which are answered as follows.

- *Sign*: Given a message m, $\mathcal{B}_2$ returns $\sigma = (\psi(v_1)^a, \psi(v_2)^b, \psi(w_2^\alpha F(M)^{a+b}), s) \leftarrow$ `Sign`(par, sk, m) but remember the random choices $a, b \leftarrow \mathbb{Z}_p$ and stores (m, σ, a, b).
- *Convert*: Given (m, σ), $\mathcal{B}_2$ checks if σ was returned as a response to a sign query m. If this is not the case, $\mathcal{B}_2$ returns $\perp$. Otherwise, $\mathcal{B}_2$ recalls the random choices a, b used to construct σ, and returns $tk_\sigma = (g_1^a, g_1^b)$.

– *Confirm/Disavow*: Given (m, σ), $\mathcal{B}_2$ simulates the confirm protocol if σ was returned as a response to a sign query on m, but simulates the disavow protocol otherwise.
  To simulate the confirm protocol, $\mathcal{B}_2$ interacts with $\mathcal{A}$ as follows. Upon receiving a commitment *com* (constructed using Σ_{Std}) and the first message a of Σ'_{Std} from $\mathcal{A}$, $\mathcal{B}_2$ choose a random challenge $c \leftarrow \mathbb{Z}_p$, computes the first message a' of $\Sigma_{Std} \vee \Sigma'_{Std}$ as if an opening of *com* is known (note that the computation of the first message a' does not require knowledge of an opening to *com*), and returns (a', c) to $\mathcal{A}$. When $\mathcal{A}$ responds with the last message z of Σ'_{Std} and a challenge c' for $\mathcal{B}_2$, $\mathcal{B}_2$ checks if (a, c, z) is an accepting transcript of Σ_{Std}. If not, $\mathcal{B}_2$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}_2$ rewinds $\mathcal{A}$ and provides $\mathcal{A}$ with a new challenge $\overline{c} \leftarrow \mathbb{Z}_p$ for the Σ'_{Std} protocol. Hence, if $\mathcal{A}$ responds with a message $\overline{z}$ such that $(a, \overline{c}, \overline{z})$ is a valid transcript for Σ'_{Std}, $\mathcal{B}_2$ can extract an opening of *com* and complete the protocol $\Sigma_{Std} \vee \Sigma'_{Std}$ honestly for any challenge c' sent by $\mathcal{A}$ (i.e. $\mathcal{B}_2$ learns a witness for Σ'_{Std} and can therefore honestly run $\Sigma_{Std} \vee \Sigma'_{Std}$). If $(a, \overline{c}, \overline{z})$ is an invalid transcript, $\mathcal{B}_2$ returns $\perp$ to $\mathcal{A}$.
  The simulation of the disavow protocol is similar to the above, except the protocols $\overline{\Sigma}'_{Std}$ and $\overline{\Sigma}_{Std} \vee \overline{\Sigma}'_{Std}$ are used.

At some stage, $\mathcal{A}$ outputs a challenge message m^*. $\mathcal{B}_2$ constructs a challenge signature by picking $s^* \leftarrow \mathbb{Z}_p$ and computing $t^* \leftarrow H_{\mathtt{k}}(\psi(u^x)||\psi(v^y)||m^*)$, $M^* \leftarrow g_1^{s^*} h_1^{t^*}$ and $\sigma^* \leftarrow (\psi(u^x), \psi(v^y), \psi(w_2^{\alpha}(h^z)^{d_0+\sum_{i=1}^n d_i M_i^*}), s^*)$, where (u^x, v^y, h^z) are the elements received in the decisional linear problem and h_1 is from *pk*. Note that if z is random, then σ^* will be a random element in $\mathbb{G}_1^3 \times \mathbb{Z}_p$, whereas if $z = x + y$, σ^* will be a valid signature on m^* since $(h^z)^{d_0+\sum_{i=1}^n d_i M_i^*} = (h^{d_0+\sum_{i=1}^n d_i M_i^*})^{x+y} = (u_0 \prod_{i=1}^n u_i^{M_i^*})^{x+y} = F(M^*)^{x+y}$.

$\mathcal{B}_2$ returns σ^* to $\mathcal{A}$ who can then ask additional sign, convert and confirm/disavow queries, but is not allowed to query σ^* to the convert or confirm/disavow oracle. $\mathcal{B}_2$ answers these queries as above. Eventually, $\mathcal{A}$ outputs a bit b which $\mathcal{B}_2$ forwards as his own solution to the decisional linear problem.

$\mathcal{B}_2$'s simulation of the invisibility experiment for $\mathcal{A}$ is perfect and it is clear that $\mathcal{B}_2$ will solve the decisional linear problem if $\mathcal{A}$ breaks the invisibility of the scheme. $\square$