On-line Non-transferable Signatures Revisited

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. We propose a new general approach to the construction of on-line non-transferable signatures introduced by Liskov and Micali. Our approach is based on an extension of designated verifier proofs which provides *interaction simulatability* as opposed to transcript simulatability. We then propose a concrete on-line non-transferable scheme which is proved secure in the standard model. Our scheme allows a less restrictive and more practical operation, is more efficient, and meets a stronger notion of security than the previous approach by Liskov and Micali.

Keywords: non-transferable signatures, standard model, provable security.

1 Introduction

An ordinary signature scheme allows anyone to verify the validity of a signature using the public key of the signer. While this property is useful in many scenarios, it might not always be desirable. For example, a signer who signs a sensitive message might prefer to be able to control who can verify the validity of his signature. Chaum *et al.* [11] addressed this with their proposal of undeniable signatures in which a verifier is required to interact with the signer to verify a signature. A signer furthermore has the possibility of proving a signature invalid, and hence in a dispute, a signer will be able to either confirm or disavow a purported signature.

However, in some scenarios, a signer might become unavailable or might refuse to cooperate with a verifier. To address this, Chaum [10] introduced designated confirmer signatures in which a third party, the confirmer, can interact with a verifier to confirm or disavow a signature on behalf of the signer. Furthermore, the confirmer can, in the case of a dispute, extract a publicly verifiable signature (of the signer) from a valid designated confirmer signature. Since their introduction, a number of undeniable schemes [9, 17, 16, 27, 23, 24] and designated confirmer schemes [29, 5, 19, 18, 40] have been proposed.

An alternative approach was proposed by Jakobsson *et al.* [22] who introduced designated verifier signatures in which only a specific verifier chosen by the signer will be convinced about the validity of a signature. This concept was extended by Steinfeld *et al.* [35] who introduced universal designated verifier signatures which allow any user (i.e. not only the signer) to convert a publicly verifiable signature into a designated verifier signature for a chosen verifier. However, these schemes do not provide a mechanism to determine the validity of a (converted) signature in a dispute. In fact, most of the proposed concrete schemes (e.g. [26, 36, 25]) enable the designated verifier to construct signatures which are perfectly indistinguishable from signatures constructed by the signer. Hence, non-repudiation cannot be enforced in these schemes. Lastly, Baek *et al.* [1] proposed another approach to universal designated verifier signatures. In their approach, any user can convert a publicly verifiable signature from a signer into a signature which cannot be verified independently. However, the user performing the conversion will be able to prove to any verifier, using an interactive protocol, that the converted signature was indeed obtained from a valid signature from the signer. Unlike

the above mentioned schemes, this does not require the verifier to hold a public/private key pair, but still cannot provide non-repudiation since there is no mechanism for proving that a converted signature does not correspond to a valid signature from the signer.

Off-line and On-line non-transferability. All of the above mentioned schemes guarantee that once a verifier has verified a signature and is convinced about its validity, he cannot transfer this conviction to a third party. This is achieved by ensuring that a verifier is able to simulate a transcript of the interaction with the signer/confirmer, and a scheme providing this property is said to be *off-line non-transferable*. However, Liskov and Micali [28] pointed out that almost all[1] previous schemes relying on interactive protocols to provide off-line non-transferability are vulnerable to *on-line* attacks, i.e. an attacker who is present *while* the verifier interacts with a signer/confirmer might be able to determine the validity of a signature by influencing messages sent by the verifier. A scheme preventing these types of attacks is said to be *on-line non-transferable* and is constructed by enabling the verifier to interactively simulate the interaction with a signer/confirmer. To preserve soundness of the scheme, only the verifier should be able to simulate a proof, and to facilitate this, it is assumed the verifier holds a public/private key pair. We note that to maintain security, it is required that the verifier knows the private key corresponding to his public key. Assuming that verifiers are not colluding with the attacker at the time of key registration, this can be achieved by letting the verifiers prove knowledge of their secret key when registering their public key. We believe this assumption is reasonable if the verifier and attacker are separate entities (note that if they are not, the attacker can trivially learn the validity of a signature simply by interacting with the signer as the verifier). We refer the reader to [28] for a discussion of this.

For undeniable signature schemes employing *non-interactive* proofs simulatable by the verifier, off-line non-transferability trivially implies on-line non-transferability. A few practical schemes with this property have been proposed in the random oracle model [23, 21], and recently Phong *et al.* [33] proposed a scheme secure in the standard model using the proof systems by Groth and Sahai [20]. Note that an undeniable signature scheme with non-interactive proofs is different from a designated verifier signature scheme in that a signature is independent of the verifier(s) and that the signer is able to disavow a signature. However, as ordinary undeniable signature schemes, these schemes rely on the signer to be on-line and cooperative in case of a dispute, since no confirmer is involved.

Furthermore, Monnerat and Vaudenay [30] proposed an undeniable signature scheme which uses 2-move confirm and disavow protocols and requires verifiers to hold a public/private key pair. Their scheme is proved secure in the random oracle model. While the used definition of non-transferability in [30] only guarantees transcript simulatability (i.e. defines off-line non-transferability), the concrete scheme allows a verifier to use his private key to simulate proofs interactively, and hence the scheme provides on-line non-transferability. However, like in the above schemes, there is no confirmer to ensure non-repudiation if the signer becomes off-line or refuses to cooperate.

Lastly, Liskov and Micali [28] proposed a generic construction of an on-line non-transferable scheme based on an `ind-cpa` secure public key encryption scheme and a `uf-cma` secure signature scheme. The scheme implements the functionality of a designated confirmer scheme and is proved secure in the standard model. Although the scheme provides a combination of functionality and security properties not provided by any previous scheme, it unfortunately

[1] The scheme by Monnerat and Vaudnay [30] mentioned below is an exception.

has a number of disadvantages. Firstly, to preserve the on-line non-transferability, a signer has to be willing to issue "fake" signatures to anyone requesting them. This is essential since a verifier will not be able to simulate a verification interaction without the ability to ask the signer for fake signatures. However, this drawback clearly limits the practical applicability of the scheme. Secondly, there is no explicit confirm protocol for the confirmer (except extraction of a publicly verifiable signature), and the confirmer can only disavow a signature through a protocol which is off-line and on-line transferable. Furthermore, unless identity-based encryption is used, a signer will have to engage in a setup process with a confirmer before signatures can be constructed. Lastly, the used unforgeability definition does not consider the possibility of corrupted confirmers.

Our contribution. We propose a new general approach to the construction of on-line non-transferable signatures with confirmers. Our approach is based on an extension of designated verifier proofs which are *interaction simulatable* i.e. a verifier is able to simulate the interaction of a proof to a third party (as opposed to just being able to simulate a transcript of the interaction). When combined with a *core signature scheme*, this property will guarantee on-line non-transferability (the core signature scheme implements the basic functionality of a designated confirmer signature scheme, see Section 4 for a formal definition).

Based on this, we propose a concrete scheme which is proved secure in the standard model[2]. The security of the scheme can be reduced to the computational Diffie-Hellman problem and the decisional linear problem. Compared to the approach taken by Liskov and Micali, our scheme has several advantages. Firstly, our scheme is proved secure in a stronger security model which does not allow a verifier to access the signer when simulating a proof of validity, and which requires that both signer and confirmer protocols are on-line non-transferable. Furthermore, our security model requires unforgeability to hold in the presence of corrupted confirmers, and guarantees that an adversary cannot impersonate either the signer or confirmer i.e. the adversary cannot make a verifier accept a validity proof, even for a valid signature, without holding the private key of the signer or confirmer. The latter security property has recently become a security requirement for undeniable signatures and designated confirmer signatures (see [24] and [40]). Lastly, our scheme does not require any setup protocol between a signer and confirmer before signatures can be generated, and our scheme employs efficient 3-move proof protocols and short signatures. In comparison, the approach by Liskov and Micali requires signatures consisting of more than $3k$ encryptions, where k is a (relatively large) security parameter, and these will be transferred to the verifier in a 4-move sign/verification protocol. Like the scheme by Liskov and Micali, our scheme is secure under concurrent attacks.

2 Preliminaries

The decisional linear problem. This decisional problem was introduced by Boneh *et al.* in [2], and is defined for a group $\mathbb{G}$ of order p as follows: Given $u, v, h, u^a, v^b, h^c \in \mathbb{G}$ for randomly chosen values $a, b \in \mathbb{Z}_p$, decide if $a+b=c$ or c is a random element of $\mathbb{Z}_p$. In [2], the decisional linear problem is shown to be computationally infeasible in the generic bilinear group model.

[2] We note that our concrete core signature scheme is constructed using somewhat similar techniques to the recently proposed undeniable signature schemes in [33], but emphasize that our construction was done independently. Furthermore, our construction supports confirmers whereas [33] does not, and unforgeability is reduced to the CDH problem whereas [33] requires the arguably non-standard q-SDH problem.

Sigma protocols. We use a description similar to [6]. 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 $(x, y) \in R$.

Trapdoor commitment schemes. We use a similar description to [15], but require perfect hiding and a perfect trapdoor property. A trapdoor commitment scheme $\mathcal{T} = \{\mathtt{G}, \mathtt{Comm}, \mathtt{TdComm}, \mathtt{TdOpen}\}$ is given by a generation algorithm $\mathtt{G}$ which, given a security parameter 1^n, returns public parameters par and a trapdoor td; a deterministic commitment algorithm $\mathtt{Comm}$ which, given par, a value $w \in \mathcal{W}$ and randomness $r \in \mathcal{R}$, returns a commitment com on w (an opening of com is simply (w, r), and a verifier checks that $com = \mathtt{Comm}(par, w, r)$); a trapdoor commitment algorithm $\mathtt{TdComm}$ that, given par, returns a commitment com' and auxiliary information aux such that the trapdoor opening algorithm $\mathtt{TdOpen}$, given aux, any value w' and the trapdoor td, returns r' such that $com' = \mathtt{Comm}(par, w', r')$. We require a trapdoor commitment scheme to have the following security properties:

- *Computational binding:* For $(par, td) \leftarrow \mathtt{G}(1^k)$, the probability that any computationally bounded adversary given par can compute (w, r, w', r') such that $w \neq w'$ and $\mathtt{Comm}(pk, w, r) = \mathtt{Comm}(pk, w', r)$, is negligible in the security parameter 1^n.
- *Perfect hiding:* For $(par, td) \leftarrow \mathtt{G}(1^n)$, random $r, r' \leftarrow \mathcal{R}$, and for any $w, w' \in \mathcal{W}$, the commitments $\mathtt{Comm}(par, w, r)$ and $\mathtt{Comm}(par, w', r')$ are distributed identically.
- *(Perfect) trapdoor property:* For $(par, td) \leftarrow \mathtt{G}(1^k)$, random $w, w' \leftarrow \mathcal{W}$, an honestly computed commitment $com \leftarrow \mathtt{Comm}(par, w, r)$ where $r \leftarrow \mathcal{R}$, and a commitment computed using the trapdoor $(com', aux) \leftarrow \mathtt{TdComm}(par)$ and $r' \leftarrow \mathtt{TdOpen}(aux, w', td)$, the values $(com, (w, r))$ and $(com', (w', r'))$ are distributed identically.

3 Designated Verifier Proofs

Designated verifier proofs were originally introduced by Jakobsson, Sako and Impagliazzo [22], but were not formally defined. Kudla and Paterson [23] formalized non-interactive designated verifier proofs and used these for the construction of a designated verifier signature scheme. In the following, we will define interactive designated verifier proofs and an extended set of security notions for these.

Informally, the aim of a proof system is to enable a prover to convince a verifier that an element e belongs to a language L. However, to make the proof systems applicable to the scenario in which we intend to use them, we will make some assumptions about L. First of

all, we assume that a language is parameterized by a pair of public keys, and we use the notation $L(pk_{P'}, pk_{P''})$ to mean the language parameterized by $(pk_{P'}, pk_{P''})$. A proof system is then defined for a family of languages $\mathcal{L}$ consisting of the languages obtained by using different public keys in the parameterization. To prove that $e \in L(pk_{P'}, pk_{P''})$ it is assumed that the prover holds the private key corresponding to either $pk_{P'}$ or $pk_{P''}$. Secondly, we will assume that elements of $L(pk_{P'}, pk_{P''})$ can be sampled using a seed s and the private key $sk_{P'}$ corresponding to $pk_{P'}$. This will be written $e \overset{s,sk_{P'}}{\leftarrow} L(pk_{P'}, pk_{P''})$ and we refer to $(pk_{P'}, sk_{P'})$ as the *primary* key pair and to $(pk_{P''}, sk_{P''})$ as the *secondary* key pair[3].

With the above assumptions, we define a designated verifier (DV) proof system $\mathcal{P}$ to consist of the following algorithms:

- `Setup`: This algorithm outputs a set of public parameters *par* which includes a description of a family of languages $\mathcal{L}$ and an element space $\mathcal{E}$.
- $\texttt{KeyGen}'_P$, $\texttt{KeyGen}''_P$, $\texttt{KeyGen}_V$: These algorithms output a primary, a secondary and a verifier public/private key pair $(pk_{P'}, sk_{P'})$, $(pk_{P''}, sk_{P''})$ and (pk_V, sk_V), respectively.
- Π: A protocol for proving that an element $e \in \mathcal{E}$ belongs to a language $L(pk_{P'}, pk_{P''}) \in \mathcal{L}$. The protocol implicitly defines a pair of interactive algorithms (`P`,`V`) run by the prover and verifier, respectively. Both algorithms take, as common input, the public keys $(pk_{P'}, pk_{P''}, pk_V)$ where pk_V is the public key of the verifier, a prover index $i_P \in \{1, 2\}$ identifying the prover key pair held by the prover, and the element e.
  - `P` takes, as additional input, the private prover key indicated by i_P i.e. $sk_{P'}$ if $i_P = 1$ and $sk_{P''}$ if $i_P = 2$. `P` interacts with `V` but has no local output.
  - `V` takes no additional input besides the common input. After completion of the interaction with `P`, `V` outputs either `accept` or `reject`.

By $\Pi\{\texttt{A} \leftrightarrow \texttt{B}\}$ we mean that the algorithms `A` and `B` interact (i.e. exchange messages) following the Π protocol. Hence, an honest run of the protocol using private prover key $sk_{P''}$ will be denoted $\Pi\{\texttt{P}(PK, i_P, e, sk_{P''}) \leftrightarrow \texttt{V}(PK, i_P, e)\}$ where $PK = (pk_{P'}, pk_{P''}, pk_V)$ and $i_P = 2$. Furthermore, we will let $z \leftarrow_2 \Pi\{\texttt{A} \leftrightarrow \texttt{B}\}$ denote the output of `B` upon completion of the protocol, and we write $sk \leftarrow_{i_P} (sk_{P'}, sk_{P''})$ to denote that $sk \leftarrow sk_{P'}$ if $i_P = 1$ and $sk \leftarrow sk_{P''}$ if $i_P = 2$.

Correctness. We require that a DV proof system is correct i.e. for all $par \leftarrow \texttt{Setup}$, all $(pk_{P'}, sk_{P'}) \leftarrow \texttt{KeyGen}'_P$, all $(pk_{P''}, sk_{P''}) \leftarrow \texttt{KeyGen}''_P$, all $i_P \in \{1, 2\}$ and corresponding $sk \leftarrow_{i_P} (sk_{P'}, sk_{P''})$, all $(pk_V, sk_V) \leftarrow \texttt{KeyGen}_V$ and all $e \in L(pk_{P'}, pk_{P''})$, the interaction $z \leftarrow_2 \Pi\{\texttt{P}(PK, i_P, e, sk) \leftrightarrow \texttt{V}(PK, i_P, e)\}$, where $PK = (pk_{P'}, pk_{P''}, pk_V)$, yields $z = \texttt{accept}$.

3.1 Security

The security notions for DV proof systems are *interaction simulatability* and *soundness* which we will define in the following. Note that we adopt multi-user security definitions in which an adversary can interact with oracles regarding a language indexed by any combination of primary and secondary keys. This property is crucial when using the DV proof systems to construct on-line non-transferable schemes which we require to be secure even when multiple signers use the same confirmer. Security models only considering a fixed signer and confirmer

[3] When using a proof system to construct an on-line non-transferable signature scheme, the primary and secondary key pair will correspond to a signer and confirmer key pair, respectively (see Section 4).

key pair might lead to schemes which are insecure in this scenario, as illustrated in [5] for designated confirmer signatures. Our multi-user security definitions will furthermore ensure security if signers use multiple confirmers. Lastly, note that in both security definitions, the adversary is allowed to run multiple instances of the prove protocol concurrently.

Interaction simulatability. A DV proof system $\mathcal{P}$ is interaction simulatable if there exists an algorithm $\mathtt{P}^{sim}$, taking input (PK, i_P, e, sk_V) where $PK = (pk_{P'}, pk_{P''}, pk_V)$ and $e \in \mathcal{E}$ is not necessarily in $L(pk_{P'}, pk_{P''})$, such that any polynomially bounded adversary $\mathcal{A}$ has negligible advantage in the experiment $\mathtt{Exp}^{int-sim}_{\mathcal{P},\mathcal{A}}(n)$ defined in Figure 1 i.e.

$$|\Pr[\mathtt{Exp}^{\text{int-sim}}_{\mathcal{P},\mathcal{A}}(n) = 1] - 1/2| < \epsilon(n)$$

where ϵ is a negligible function of the security parameter n. In the experiment, the function $l(n)$ represents the number of users in the scheme which is assumed to be polynomial in n. Furthermore, $\mathcal{A}$ has access to the oracles $\mathcal{O} = \{Corrupt, EGen, Prove\}$ which are defined below. $Q_{P'}$ and $Q_{P''}$ represent lists of corrupted primary and secondary key pairs, respectively, and will be updated in corrupt queries.

- *Corrupt*: given (i, P'), (j, P'') or (k, V), this oracle adds i to $Q_{P'}$ or j to $Q_{P''}$ if the query is one of the first two types. Then the oracle returns $sk^i_{P'}$, $sk^j_{P''}$ or sk^k_V, depending on the query type.
- *EGen*: given a seed s and indices (i, j), this oracle returns an element $e \overset{s, sk^i_{P'}}{\leftarrow} L(pk^i_{P'}, pk^j_{P''})$.
- *Prove*: given $e \in \mathcal{E}$, indices (i, j, k) and a prover index i_P, this oracle interacts with $\mathcal{A}$ by setting $PK \leftarrow (pk^i_{P'}, pk^j_{P''}, pk^k_V)$, setting $sk \leftarrow_{i_P} (sk^i_{P'}, sk^j_{P''})$, and running $\mathtt{P}(PK, i_P, e, sk)$ if $e \in L(pk^i_{P'}, pk^j_{P''})$. If $e \notin L(pk^i_{P'}, pk^j_{P''})$, the oracle returns $\perp$.

It is required that the challenge prover keys are uncompromised i.e. $i^* \notin Q_{P'}$ and $j^* \notin Q_{P''}$, and $\mathcal{A}$ is not allowed to submit the challenge element e^* to the *Prove* oracle in the experiment. Note that $\mathcal{A}$ can obtain simulated proofs for $e \in \mathcal{E}$ by running $\mathtt{P}^{sim}(PK, i_P, e, sk_V)$ himself for any public keys $PK = (pk^i_{P'}, pk^j_{P''}, pk^k_V)$, since he is allowed to corrupt any private verifier key sk_V.

Soundness. For a proof system $\mathcal{P}$, soundness is defined via the experiment $\mathtt{Exp}^{soundness}_{\mathcal{P},\mathcal{A}}(n)$ shown in Figure 1 where $Q_{P'}$, $Q_{P''}$ and Q_V are lists of corrupted primary, secondary and verifier key pairs, respectively. In the experiment, the adversary has access to the oracles $\mathcal{O} = \{Corrupt, EGen, Prove, Sim\}$ which are defined as follows:

- *Corrupt*, *EGen*, *Prove*: defined as above in the interaction simulatability experiment, except that the *Corrupt* oracle now maintains the additional list Q_V of corrupted verifiers.
- *Sim*: given $e \in \mathcal{E}$, indices (i, j, k) and a prover index i_P, this oracle interacts with $\mathcal{A}$ by constructing $PK \leftarrow (pk^i_{P'}, pk^j_{P''}, pk^k_V)$ and running $\mathtt{P}^{sim}(PK, i_P, e, sk^j_V)$.

A proof system is sound if for any computationally bounded adversary $\mathcal{A}$ we have that

$$\Pr[\mathtt{Exp}^{\text{soundness}}_{\mathcal{P},\mathcal{A}}(n) = 1] < \epsilon(n)$$

Note that this soundness definition not only captures that a prover should not be able to make a verifier accept a proof for an element $e \notin L(pk_{P'}, pk_{P''})$, but also requires that a prover not knowing the relevant private key ($sk_{P'}$ if $i_P = 1$ or $sk_{P''}$ if $i_P = 2$) cannot make a verifier accept, even for elements $e \in L(pk_{P'}, pk_{P''})$. The latter security property is referred to as security against impersonation attacks, and has recently been defined for undeniable signatures [24] and designated confirmer signatures [40].

$\mathtt{Exp}^{\text{int-sim}}_{\mathcal{P},\mathcal{A}}(n)$
$Q_{P'} \leftarrow \emptyset$, $Q_{P''} \leftarrow \emptyset$
$par \leftarrow \mathtt{Setup}(1^n)$
For $i = 1, \ldots, l(n)$
$(pk^i_{P'}, sk^i_{P'}) \leftarrow \mathtt{KeyGen}'_P(par)$
$(pk^i_{P''}, sk^i_{P''}) \leftarrow \mathtt{KeyGen}''_P(par)$
$(pk^i_V, sk^i_V) \leftarrow \mathtt{KeyGen}_V(par)$
$(i^*, j^*, k^*, i^*_P, s^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk^1_{P'}, \ldots, pk^l_V)$
$PK^* \leftarrow (pk^{i^*}_{P'}, pk^{j^*}_{P''}, pk^{k^*}_V)$
$sk^* \leftarrow_{iP} \{sk^{i^*}_{P'}, sk^{j^*}_{P''}\}$
$b \leftarrow \{0, 1\}$
If $b = 0$
$e^* \overset{s^*, sk^{i^*}_{P'}}{\leftarrow} L(pk^{i^*}_{P'}, pk^{j^*}_{P''})$
$b' \leftarrow_2 \Pi\{\mathtt{P}(PK^*, i^*_P, e^*, sk^*) \leftrightarrow \mathcal{A}^{\mathcal{O}}(st_{\mathcal{A}}, e^*)\}$
else
$e^* \overset{\$}{\leftarrow} \mathcal{E}$
$b' \leftarrow_2 \Pi\{\mathtt{P}^{sim}(PK^*, i^*_P, e^*, sk^{k^*}_V) \leftrightarrow \mathcal{A}^{\mathcal{O}}(st_{\mathcal{A}}, e^*)\}$
If $b = b'$ return 1
else return 0

$\mathtt{Exp}^{\text{soundness}}_{\mathcal{P},\mathcal{A}}(n)$
$Q_{P'} \leftarrow \emptyset$, $Q_{P''} \leftarrow \emptyset$, $Q_V \leftarrow \emptyset$
$par \leftarrow \mathtt{Setup}(1^n)$
For $i = 1, \ldots, l(n)$
$(pk^i_{P'}, sk^i_{P'}) \leftarrow \mathtt{KeyGen}'_P(par)$
$(pk^i_{P''}, sk^i_{P''}) \leftarrow \mathtt{KeyGen}''_P(par)$
$(pk^i_V, sk^i_V) \leftarrow \mathtt{KeyGen}_V(par)$
$(i^*, j^*, k^*, i^*_P, e^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk^1_{P'}, \ldots, pk^l_V)$
$PK^* \leftarrow (pk^{i^*}_{P'}, pk^{j^*}_{P''}, pk^{k^*}_V)$
$z \leftarrow_2 \Pi\{\mathcal{A}^{\mathcal{O}}(st_{\mathcal{A}}) \leftrightarrow \mathtt{V}(PK^*, i^*_P, e^*)\}$
If $z = \mathtt{accept} \wedge k^* \notin Q_V \wedge \big(e^* \notin L(pk^{i^*}_{P'}, pk^{j^*}_{P''}) \vee$
$(i_P = 1 \wedge i^* \notin Q_{P'}) \vee (i^*_P = 2 \wedge j^* \notin Q_{P''})\big)$
return 1
else return 0

Fig. 1. Interaction simulatability and soundness experiments

4 On-line Non-transferable Signatures

An on-line non-transferable signature scheme is defined in a similar way to a designated confirmer signature (DCS) scheme, and involves a signer S, a verifier V and a confirmer C. However, we consider a scheme to consist of two parts: a *core signature scheme* and a set of DV proof systems. This logical separation is inspired by the construction of designated verifier signatures by Kudla and Paterson [23], and will be useful when establishing the general security results in Section 5. A core signature scheme is given by the following algorithms:

- $\mathtt{Setup}$: Returns a set of public parameters par, including a public/private key space for signers $(\mathcal{PK}_S, \mathcal{SK}_S)$ and confirmers $(\mathcal{PK}_C, \mathcal{SK}_C)$.
- $\mathtt{KeyGen}_S$, $\mathtt{KeyGen}_C$: Given input par, these algorithms generate a public/private key pair for a signer (pk_S, sk_S) and a confirmer (pk_C, sk_C), respectively.
- $\mathtt{Sign}$: Given input par, a message m, a private signer key sk_S and a public confirmer key pk_C, this algorithm outputs a signature σ on m.
- $\mathtt{Extract}_C$: Given input par, a message m, a public signer key pk_S, a secret confirmer key sk_C and a signature $\sigma \in \{\mathtt{Sign}(par, m, sk_S, pk_C)\}$, where sk_S is the private key corresponding to pk_S, this algorithm outputs an extracted signature σ' on m under pk_S.
- $\mathtt{Verify}$: Given input par, a message m, a public signer key pk_S and a purported extracted signature σ', this algorithm outputs either $\mathtt{accept}$ or $\mathtt{reject}$.

For given public signer and confirmer keys pk_S and pk_C, the $\mathtt{Sign}$ algorithm of the core signature scheme defines a language consisting of valid message/signature pairs i.e. $L(pk_S, pk_C) = \{(m, \sigma) : \sigma \leftarrow \mathtt{Sign}(par, m, sk_S, pk_C)\}$ where sk_S is the private key corresponding to pk_S. The complement language $\overline{L(pk_S, pk_C)}$ consists of the message/signature pairs (m, σ) for which σ is not a valid signature on m under pk_S and pk_C. In the following, we will consider the families of languages $\mathcal{L} = \{L(pk_S, pk_C) : pk_S \in \mathcal{PK}_S, pk_C \in \mathcal{PK}_C\}$ and $\overline{\mathcal{L}} = \{\overline{L(pk_S, pk_C)} : pk_S \in \mathcal{PK}_S, pk_C \in \mathcal{PK}_C\}$.

An on-line non-transferable signature scheme defines DV proof systems $\mathcal{P}$ and $\overline{\mathcal{P}}$ for $\mathcal{L}$ and $\overline{\mathcal{L}}$ i.e. for proving validity and invalidity of signatures. Note that the primary and secondary keys of $\mathcal{P}$ and $\overline{\mathcal{P}}$ correspond to the signer and confirmer keys for the core signature scheme, respectively. It is assumed that $\mathcal{P}$ and $\overline{\mathcal{P}}$ use the same type of verifier keys and share a common $\texttt{KeyGen}_V$ algorithm, and that any public parameters required by the proof systems are output together with the public parameters *par* of the core signature scheme.

Let $\Pi = (\texttt{P}, \texttt{V})$ and $\overline{\Pi} = (\overline{\texttt{P}}, \overline{\texttt{V}})$ denote the proof protocols for $\mathcal{P}$ and $\overline{\mathcal{P}}$. The DV proof systems implement the following algorithm and protocols:

- $\texttt{KeyGen}_V$: This algorithm outputs a public/private verifier key pair (pk_V, sk_V).
- $\texttt{Confirm}_S$, $\texttt{Confirm}_C$: Both protocols correspond to running Π, but $i_P = 1$ for $\texttt{Confirm}_S$ and $i_P = 2$ for $\texttt{Confirm}_C$. The prover (the signer or the confirmer) runs $\texttt{P}$ with private input $sk \leftarrow_{i_P} (sk_S, sk_C)$, and the verifier runs the corresponding algorithm $\texttt{V}$. The common input is $(PK, i_P, (m, \sigma))$ where $PK = (pk_S, pk_C, pk_V)$.
- $\texttt{Disavow}_S$, $\texttt{Disavow}_C$: Similar to the above, both protocols correspond to running $\overline{\Pi}$. The common and private inputs are as in the confirm protocols.

With the above algorithms and protocols, an on-line non-transferable signature scheme is defined to be the set:

$$\mathcal{S} = \{\texttt{Setup}, \texttt{KeyGen}_S, \texttt{KeyGen}_C, \texttt{KeyGen}_V, \texttt{Sign}, \\ \texttt{Confirm}_S, \texttt{Confirm}_C, \texttt{Disavow}_S, \texttt{Disavow}_C, \texttt{Extract}_C, \texttt{Verify}\}$$

Note that the above description requires that both the signer and confirmer can confirm and disavow a signature. However, this might not be needed in many practical scenarios, and, in comparison, many DCS schemes (e.g. [29, 5, 19, 18]) are defined for a restricted functionality in which only the confirmer can disavow a signature. In the above description, this corresponds to using a proof system $\overline{\mathcal{P}}$ which only supports proofs where $i_P = 2$. Some schemes (e.g. [38] and Liskov and Micali's scheme) furthermore restrict the functionality by only requiring that the signer proves validity of a signature at the time of signing i.e. $\texttt{Confirm}_S$ and $\texttt{Sign}$ are combined into a single interactive protocol $\texttt{ConfirmedSign}$ in which the verifier obtains a signature and will be convinced about its validity, and the signer is not required to be able to prove validity of a signature at a later stage. For the general results presented in Section 5, we will consider all three types of schemes, but we will restrict the last type to schemes in which a signer can generate a signature independently of the verifier i.e. $\texttt{ConfirmedSign}$ consists of two stages: signature generation performed only by the signer, followed by an interactive proof of validity. However, we allow a signer to make use of the randomness from the signature generation stage as an additional private input in the interactive proof of validity. For such schemes, the language $\mathcal{L}$ is defined using a modified $\texttt{ConfirmedSign}$ which only executes the signature generation stage and terminates.

The security definitions in Section 3.1 and Section 4.1 can easily be modified to model the security of these schemes by restricting the type of oracle queries an adversary can make.

4.1 Security of Core Signature Scheme

Unforgeability. For a core signature scheme $\mathcal{S}$, unforgeability is defined via the experiment $\texttt{Exp}^{\text{uf-cma}}_{\mathcal{S},\mathcal{A}}(n)$ shown in Figure 2, where Q_S denotes the set of compromised signers and Q_σ denotes the set of tuples (i, j, m, σ) where σ was returned by the signing oracle on a request for a signature on m under (pk^i_S, pk^j_C). In the experiment $\mathcal{A}$ has access to the following oracles:

$\mathtt{Exp}^{\text{uf-cma}}_{\mathcal{S},\mathcal{A}}(n)$
$Q_S \leftarrow \emptyset$, $Q_\sigma \leftarrow \emptyset$
$par \leftarrow \mathtt{Setup}(1^n)$
For $i = 1, \ldots, l(n)$
$(pk_S^i, sk_S^i) \leftarrow \mathtt{KeyGen}_S(par)$
$(pk_C^i, sk_C^i) \leftarrow \mathtt{KeyGen}_C(par)$
$(pk_V^i, sk_V^i) \leftarrow \mathtt{KeyGen}_V(par)$
$(i^*, j^*, m^*, \sigma^*)/(i', m', \sigma') \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk_S^1, \ldots, pk_V^l)$
// *forgery of ordinary signature*
If $i^* \notin Q_S \wedge (i^*, j^*, m^*, \sigma^*) \notin Q_\sigma \wedge$
$(m^*, \sigma^*) \in L(pk_S^{i^*}, pk_C^{j^*})$
return 1
// *forgery of extracted signature*
$z \leftarrow \mathtt{Verify}(pk_S^{i'}, m', \sigma')$
If $i' \notin Q_S \wedge (i', -, m', -) \notin Q_\sigma \wedge z = \mathtt{accept}$
return 1
else return 0

$\mathtt{Exp}^{\text{inv-cma}}_{\mathcal{S},\mathcal{A}}(n)$
$Q_S \leftarrow \emptyset$, $Q_C \leftarrow \emptyset$
$par \leftarrow \mathtt{Setup}(1^n)$
For $i = 1, \ldots, l(n)$
$(pk_S^i, sk_S^i) \leftarrow \mathtt{KeyGen}_S(par)$
$(pk_C^i, sk_V^i) \leftarrow \mathtt{KeyGen}_C(par)$
$(pk_V^i, sk_V^i) \leftarrow \mathtt{KeyGen}_V(par)$
$(i^*, j^*, m^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk_S^1, \ldots, pk_V^l)$
$b \xleftarrow{\$} \{0, 1\}$
If $b = 0$ $\sigma^* \xleftarrow{\$} \mathcal{S}_\sigma$
else $\sigma^* \leftarrow \mathtt{Sign}(par, m^*, sk_S^{i^*}, pk_C^{j^*})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(st_{\mathcal{A}}, \sigma^*)$
If $b = b'$
return 1
else return 0

Fig. 2. Unforgeability and invisibility experiments

- *Corrupt*: Given (i, S), (j, C) or (k, V), this oracle adds i to Q_S in the first case, and returns sk_S^i, sk_C^j or sk_V^k, depending on the input.
- *Sign*: Given a signer index i, a confirmer index j and a message m, this oracle computes $\sigma \leftarrow \mathtt{Sign}(par, pk_C^j, sk_S^i, m)$, adds (i, j, m, σ) to Q_σ and returns σ.
- *Confirm*$_S$, *Confirm*$_C$: Given indices i, j, k, a prover index i_P, a message m and a signature σ, this oracle returns $\perp$ if $(m, \sigma) \notin L(pk_S^i, pk_C^j)$. Otherwise, the oracle interacts with $\mathcal{A}$ by setting $PK \leftarrow (pk_S^i, pk_C^j, pk_V^k)$, $sk \leftarrow_{i_P} (sk_S^i, sk_C^j)$ and running $\mathtt{P}(PK, i_P, (m, \sigma), sk)$.
- *Disavow*$_S$, *Disavow*$_C$: Given indices i, j, k, a prover index i_P, a message m and a signature σ, this oracle returns $\perp$ if $(m, \sigma) \in L(pk_S^i, pk_C^j)$. Otherwise, the oracle interacts with $\mathcal{A}$ by setting $PK \leftarrow (pk_S^i, pk_C^j, pk_V^k)$, $sk \leftarrow_{i_P} (sk_S^i, sk_C^j)$ and running $\overline{\mathtt{P}}(PK, i_P, (m, \sigma), sk)$.

Since $\mathcal{A}$ is allowed to corrupt all confirmers and verifiers while trying to construct a forgery, he can extract signatures and run simulated versions of the confirm and disavow protocols by himself, and it is not needed to provide $\mathcal{A}$ with oracles for these tasks. A scheme is said to be unforgeable if

$$\Pr[\mathtt{Exp}^{\text{uf-cma}}_{\mathcal{S},\mathcal{A}}(n) = 1] < \epsilon(n).$$

Note that an extracted signature is not tied to a confirmer, and hence, for a forgery of an extracted signature, it is required that $\mathcal{A}$ did not submit the sign query (i^*, j, m^*) for any confirmer index j.

A scheme which is secure against adversaries not making any confirm or disavow queries is said to be *unforgeable against passive adversaries*. In case a scheme combines `Sign` and $\mathtt{Confirm}_S$ in a single protocol `ConfirmedSign`, we use a modified version of `ConfirmedSign` which only generates a signature and halts, when considering passive adversaries. When distinction is required, we refer to non-passive adversaries as *active* adversaries.

Invisibility. For a core signature scheme $\mathcal{S}$, invisibility is defined via the experiment $\mathtt{Exp}^{\text{inv-cma}}_{\mathcal{S},\mathcal{A}}(n)$ shown in Figure 2, where Q_S and Q_C are the sets of compromised signers and confirmers, respectively, and $\mathcal{S}_\sigma$ is the signature space. In the experiment, $\mathcal{A}$ has access to the same oracles as in the unforgeability game, but is also given access to an additional extraction oracle:

– *Extract*: Given indices i, j and a message/signature pair $(m, \sigma) \in L(pk_S^i, pk_C^j)$, the oracle returns the extracted signature $\sigma' \leftarrow \texttt{Extract}(pk_S^i, sk_C^j, m, \sigma)$. If $(m, \sigma) \notin L(pk_S^i, pk_C^j)$, the oracle returns $\perp$.

Furthermore, $\mathcal{A}$ is not allowed to compromise the challenge signer or prover (i.e. it is required that $i^* \notin Q_S$, $j^* \notin Q_C$), or submit the received challenge signature σ^* to the *Confirm*, *Disavow* or *Extract* oracles. A scheme is invisible if

$$|\Pr[\texttt{Exp}^{\text{inv-cma}}_{\mathcal{S},\mathcal{A}}(n) = 1] - 1/2| < \epsilon(n)$$

Invisibility against *passive* adversaries is defined in a similar way to unforgeability against passive adversaries.

4.2 Security of Combined Scheme

Ideally, when combining a secure (i.e. unforgeable and invisible) core signature scheme with a set of secure (i.e. interaction simulatable and sound) proof systems, the proof systems would maintain their security properties. However, this is not obvious since a potential adversary will have access to additional information such as extracted signatures and proofs of elements not belonging to a language.

To address this concern, we define an extension of the previously given security notions for a proof system $\mathcal{P}$. We now assume that the language $\mathcal{L}$ is defined by a core signature scheme and that a corresponding proof system $\overline{\mathcal{P}}$ for $\overline{\mathcal{L}}$ with a prove protocol $\overline{\Pi} = (\overline{\mathsf{P}}, \overline{\mathsf{V}})$ is defined. For this proof system we consider the oracles $\overline{\mathit{Prove}}$ and $\overline{\mathit{Sim}}$ which are defined in a similar way to the oracles *Prove* and *Sim* in the soundness experiment, but for $\overline{\mathcal{L}}$. We additionally consider the oracle *Extract* defined as in the invisibility experiment.

Definition 1 *A proof system $\mathcal{P}$ is said to be interaction simulatable (sound) against an adversary with full oracle access, if the proof system is secure according to the definition in Section 3.1 while the adversary has access to the additional oracles $\{\overline{\mathit{Prove}}, \overline{\mathit{Sim}}, \mathit{Extract}\}$.*

In the interaction simulatability experiment, the adversary is not allowed to submit the challenge element to the $\overline{\mathit{Prove}}$ or *Extract* oracle. (Note that the definition is also applicable when $\mathcal{P}$ implements the disavow protocol.)

We say that an on-line non-transferable signature scheme is *secure* if it consists of an unforgeable and invisible core signature scheme, and the confirm and disavow protocols are implemented with DV proof systems which are interaction simulatable and sound against adversaries with full oracle access. Note that this interpretation of on-line non-transferability is stronger than that of Liskov and Micali [28] in that both the confirm and disavow protocols are required to be interaction simulatable and a verifier is required to be able to simulate a proof without access to the signer.

5 Construction of DV Proof Systems

Our approach to the construction of DV proof systems is inspired by the original approach by Jakobsson *et al.* [22] and is both simple and intuitive. The approach is furthermore closely related to the construction of efficient zero-knowledge proofs in the auxiliary string model [15]. The proof systems are based on sigma protocols which are modified using a trapdoor commitment scheme to make them usable in a designated verifier setting. More specifically, a prover and a verifier interact as follows.

1. The prover chooses a random value w from the challenge space of the sigma protocol, computes $com \leftarrow \texttt{Comm}(par, w, r)$ for random r, and sends com together with the first message a of the sigma protocol to the verifier.
2. The verifier then sends a random challenge c to the prover.
3. The prover uses $c + w$ as a challenge instead of c and sends the opening (w, r) together with the last message z of the sigma protocol to the verifier. The verifier checks that w is in the challenge space for the sigma protocol, checks that $com = \texttt{Comm}(par, w, r)$, and accepts if $(a, c + w, z)$ is an accepting transcript of the sigma protocol.

It is assumed that the commitment scheme allows commitments to all possible challenge values of the sigma protocol. The parameters and trapdoor (par, td) of the commitment scheme will be used as the public/private key pair (pk_V, sk_V) of the verifier. Hence, using sk_V, the verifier will be able to use a predetermined challenge value in the above interaction, and will thus have the ability to simulate the proof interactively.

However, this is not sufficient for proving our constructions secure. Essentially the problem is that an adversary can request to interact with $\texttt{P}^{sim}$ in the soundness experiment, choosing any element as input (even elements $e \notin L(pk_S, pk_C)$). This type of query can be difficult to handle for a simulator not knowing the trapdoor of the commitment scheme, whereas a simulator knowing the trapdoor might not gain sufficient information from an adversary breaking the security of the proof system.

To overcome this problem, we introduce a stronger binding property for the used commitment scheme. Specifically, we require that for any computationally bounded algorithm $\mathcal{A}$, the probability

$$\Pr[(par, td) \leftarrow \texttt{G}(1^k); (w, r, w', r') \leftarrow \mathcal{A}^{\mathcal{O}_c, \mathcal{O}_o}(par) : w \neq w' \wedge \texttt{Comm}(par, w, r) = \texttt{Comm}(par, w', r')]$$

is negligible in the security parameter n when $\mathcal{A}$ has access to a commit and an open oracle, $\mathcal{O}_c$ and $\mathcal{O}_o$, which behave as follows: upon request, $\mathcal{O}_c$ computes $(com, aux) \leftarrow \texttt{TdComm}$, stores aux and returns com to $\mathcal{A}$. Given a commitment com returned by $\mathcal{O}_c$ and a value w, $\mathcal{O}_o$ retrieves aux and returns $r \leftarrow \texttt{TdOpen}(aux, w, td)$ i.e. $com = \texttt{Comm}(par, w, r)$. The adversary is only allowed to query $\mathcal{O}_o$ with a commitment com obtained from O_c, and is not allowed to make more than one query to $\mathcal{O}_o$ for a given commitment com.

Informally, this enhanced security notion requires that the commitment scheme is binding even though the adversary is allowed to choose how a polynomial number of commitments should be opened, after seeing these commitments. We say that a trapdoor commitment scheme is *binding under selective trapdoor openings* if it satisfies the above property. Commitment schemes providing this type of security will play an important role in our security proofs.

It is fairly easy to show that Pedersen's commitment scheme [32] is binding under selective trapdoor openings assuming the one more discrete logarithm problem is hard. However, to obtain a commitment scheme which can be shown secure only assuming the ordinary discrete logarithm problem is hard, we can make use of the following "double trapdoor" extension also used to strengthen signatures [37] and improve on-line/off-line signatures [4, 8]:

- $\texttt{G}$: Given 1^n, pick a group $\mathbb{G}$ of prime order p such that $2^n < p < 2^{n+1}$, and furthermore pick a generator $g \leftarrow \mathbb{G}$ and random $x_1, x_2 \leftarrow \mathbb{Z}_p$. Compute $h_1 \leftarrow g^{x_1}$ and $h_2 \leftarrow g^{x_2}$, and set the public parameters to $par \leftarrow (\mathbb{G}, g, h_1, h_2)$ and the trapdoor to $td \leftarrow (x_1, x_2)$.
- $\texttt{Comm}$: Given a value $w \in \mathbb{Z}_p$, pick random $r_1, r_2 \leftarrow \mathbb{Z}_p$ and compute a commitment to w as $com \leftarrow g^w h_1^{r_1} h_2^{r_2}$. (The opening of com is (w, r_1, r_2).)

– `TdComm`: Pick random $r \leftarrow \mathbb{Z}_p$ and set $com \leftarrow g^r$ and $aux \leftarrow r$.
– `TdOpen`: Given com, a corresponding aux, the trapdoor $td = (x_1, x_2)$ and a value w, pick random $r_1 \leftarrow \mathbb{Z}_p$, compute $r_2 \leftarrow (aux - w - x_1 r_1)/x_2$, and return the opening (w, r_1, r_2)

Theorem 2 *Assume the discrete logarithm problem is hard in $\mathbb{G}$. Then the above commitment scheme is binding under selective trapdoor openings.*

The proof of this theorem is relatively straight forward, and we omit the details here.

5.1 Security

By making a few assumptions about the core signature scheme and how the DV proof systems are constructed, we can obtain some general security results which will greatly simplify the security proofs of our concrete constructions. More specifically, let $\mathcal{S}$ be a core signature scheme. We assume that, given a public confirmer key pk_C of $\mathcal{S}$, it is infeasible to compute the corresponding private key, even with access to a signature validity oracle which, given a public/private signer key pair (pk_S, sk_S), a message m and a signature σ, returns 1 if σ is a valid signature on m under (pk_S, pk_C), and returns 0 otherwise. Note that this oracle is trivial to implement if the validity of a signature can be determined knowing only sk_S, and in this case, the assumption is equivalent to assuming that a private confirmer key cannot be computed from the corresponding public key. However, for schemes in which knowledge of the randomness used to construct a signature is required for the signer to prove validity, sk_S alone might not be sufficient to determine the validity of a signature, and the validity oracle might provide non-trivial information[4]. We will refer to this property as *extended key security for confirmers*.

We will furthermore make some assumptions about how the DV proof systems are constructed. Let $\mathcal{L}$ be the family of languages of valid signatures indexed by a public signer and confirmer key, and assume that Σ' and Σ'' are sigma protocols with the following properties:

– Given common input (pk_S, pk_C, m, σ), both Σ' and Σ'' prove the validity of (m, σ) under (pk_S, pk_C).
– The private prover input in Σ' is a value x such that (x, m, σ) can be used to construct a selective forgery on any message m'.
– The private prover input in Σ'' is the private confirmer key sk_C.

It is assumed that Σ' and Σ'' have special soundness i.e. the private prover input specified above can be extracted from two transcripts with the same commitment message but different challenge messages, and Σ' and Σ'' are furthermore assumed to be special honest verifier zero-knowledge. Then let DV-Σ' and DV-Σ'' be the protocols obtained from modifying Σ' and Σ'' as described in the previous section, using a trapdoor commitment scheme $\mathcal{T}$ which is binding under selective trapdoor openings, is perfectly hiding and has a perfect trapdoor property[5]. Finally, let $\mathcal{P}$ be a proof system for the family of languages $\mathcal{L}$ using a proof protocol Π that executes DV-Σ' if $i_P = 1$ and executes DV-Σ'' if $i_P = 2$. Similarly, let $\overline{\mathcal{P}}$ be a proof system for $\overline{\mathcal{L}}$, constructed using the sigma protocols $\overline{\Sigma}'$ and $\overline{\Sigma}''$ with similar properties to Σ' and

[4] Note that since it is required that a confirmer can prove the validity of a signature without any special knowledge besides his private key, it follows that sk_C alone is sufficient to determine validity of a signature.
[5] We assume that given the same security parameter 1^n as the core signature scheme, the trapdoor commitment scheme will generate (par, td) that allows commitments to values in the challenge space of Σ' and Σ''.

Σ'', except that they prove the invalidity of a signature. Then the following results can be obtained.

Theorem 3 *Assume that $\mathcal{S}$ is invisible against passive adversaries. Then $\mathcal{P}$ ($\overline{\mathcal{P}}$) constructed above is an interaction simulatable DV proof system for $\mathcal{L}$ ($\overline{\mathcal{L}}$).*

Theorem 4 *Assume that $\mathcal{S}$ is unforgeable against passive adversaries and provides extended key security for confirmers. Then $\mathcal{P}$ ($\overline{\mathcal{P}}$) constructed above is a sound DV proof system for $\mathcal{L}$ ($\overline{\mathcal{L}}$).*

The proofs of the above theorems can be found in Appendices A and B, respectively. Note that although the proofs of the above theorems consider the most general type of on-line non-transferable signature schemes, it is easy to verify that the proofs are applicable for schemes which implement a more limited functionality as discussed in Section 4. Especially note that the simulator in the proofs is not assumed to be able to determine validity of a signature using sk_S alone, and is not required to interact in ordinary proofs with $i_P = 1$ for signatures not constructed by the simulator himself, making it easy to verify that the proofs are valid for the third type of scheme discussed in Section 4.

The above theorems show that the constructed DV proof systems are secure when considering the proof systems themselves, but do not guarantee security when the adversary has access to additional information provided by a combined scheme as discussed in Section 4.2. However, Theorem 3 can easily be extended to the following results.

Theorem 5 *Assume that $\mathcal{S}$ is invisible against passive adversaries. Then $\mathcal{P}$ and $\overline{\mathcal{P}}$ constructed above are interaction simulatable against adversaries with full oracle access.*

Likewise, it is possible to extend Theorem 4, but to handle extract queries correctly, an additional assumption is needed. More specifically, we assume there exists an algorithm `SimExtract` that given the private signer key sk_S and a *valid* signature $(m, \sigma) \in L(pk_S, pk_C)$, computes an extracted signature $\hat{\sigma} \leftarrow \texttt{SimExtract}(pk_S, pk_C, sk_S, (m, \sigma))$ such that $\hat{\sigma} = \sigma' \leftarrow \texttt{Extract}(pk_S, pk_C, sk_C, (m, \sigma))$ i.e. extracted signatures can be computed knowing sk_S[6]. With this assumption, the following theorem can be obtained.

Theorem 6 *Assume that $\mathcal{S}$ is unforgeable against passive adversaries and provides extended key security for confirmers, and the above assumption holds about $\mathcal{S}$. Then $\mathcal{P}$ and $\overline{\mathcal{P}}$ are sound against adversaries with full oracle access.*

Given the proofs of Theorem 3 and 4, the proofs of the above theorems are relatively easy extensions (using the additional assumption for Theorem 6), and are omitted here.

The techniques used in the proof for Theorem 3 and 4 can furthermore be used to extend the security for the core signature scheme. As above, this extension is fairly simple, and the proofs of the following theorems will be omitted.

Theorem 7 *Assume that $\mathcal{S}$ is unforgeable against passive adversaries, and that $\mathcal{P}$ and $\overline{\mathcal{P}}$ constructed above implement the `Confirm` and `Disavow` protocols. Then $\mathcal{S}$ is unforgeable against active adversaries.*

[6] Note that this assumption does not necessarily mean that the validity of a signature can be determined using sk_S, since a valid extracted signature might also be obtained from an invalid original signature when extraction is done using sk_S.

Theorem 8 *Assume that $\mathcal{S}$ is unforgeable and invisible against passive adversaries, and that $\mathcal{P}$ and $\overline{\mathcal{P}}$ constructed above implement the* `Confirm` *and* `Disavow` *protocols. Then $\mathcal{S}$ is invisible against active adversaries.*

Like Theorems 3 and 4, it is fairly easy to verify that similar results to Theorems 5, 6, 7 and 8 hold for schemes implementing a restricted functionality.

With the above theorems, the task of constructing an on-line non-transferable signature scheme has been reduced to the simpler task of constructing a passively secure core signature schemes in which validity and invalidity of a signature can be proved using sigma protocols with the above stated properties.

6 Concrete On-line Non-transferable Scheme

Our concrete on-line non-transferable signature scheme is based on an adaptation of Waters' signatures [39] combined with the technique by Boneh, Shen and Waters [3]. The scheme belongs to the third type of scheme discussed in Section 4 in that only the confirmer is able to disavow a signature, and `Sign` and $\texttt{Confirm}_S$ are implemented in a single `ConfirmedSign` protocol.

In our description of the confirm and disavow protocols, we will make use of the notation $\Sigma\{(x,y) : g^x = h \wedge u^y = j \wedge v^{x+y} = k\}$ to denote a sigma protocol in which the prover gets the private prover input (x,y) and proves to a verifier that the equalities $g^x = h$, $u^y = j$ and $u^x v^y = k$ hold for group elements $g, u, h, j \in \mathbb{G}$ and $v, k \in \mathbb{G}'$ ($\mathbb{G}$ and $\mathbb{G}'$ might be different groups of the same order). It is well known that such sigma protocols can be obtained by combining multiple instances of Schnorr's protocol for proving knowledge of a discrete logarithm [7]. Furthermore, we will make use of a sigma protocol of the form $\Sigma\{(x,y) : u^x = g \wedge v^y = g \wedge a^x b^y \neq c\}$. Such a protocol can be implemented using a similar technique to the protocol for proving in-equality of discrete logarithms by Camenish and Shoup [6] i.e. a prover and verifier interact as follows:

- The signer picks random r, computes $C \leftarrow (a^x b^y / c)^r$ and sends C to the verifier.
- The signer and verifier then interact in the protocol $\Sigma\{(\alpha,\beta,\gamma) : u^\alpha g^{-\gamma} = 1 \wedge v^\beta g^{-\gamma} = 1 \wedge a^\alpha\ b^\beta c^{-\gamma} = C\}$ where $\alpha = xr$, $\beta = yr$ and $\gamma = r$.

The value C can be sent together with the first message of the sigma protocol in the second step to obtain a 3-move protocol. To see that the above protocol is special honest verifier zero-knowledge and has special soundness, similar arguments to [6] can be used, and we refer the reader to [6] for the details. Lastly we assume that a trapdoor commitment scheme $\mathcal{T} = (\texttt{G}, \texttt{Comm}, \texttt{TdComm}, \texttt{TdOpen})$ is given[7], and we let DV-Σ denote the sigma protocol Σ modified as described in Section 5 using $\mathcal{T}$.

Our concrete on-line non-transferable signature scheme $\mathcal{S}$ is defined as follows:

- `Setup`: Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p, a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ and generator g of $\mathbb{G}$. Furthermore, choose a collision resistant hash function family $\mathcal{H} = \{H_{\texttt{k}} : \{0,1\}^* \rightarrow \mathbb{Z}_p\}$ indexed by a key $\texttt{k} \in \mathcal{K}$. Return $par = (\mathbb{G}, \mathbb{G}_T, e, p, g, \mathcal{H})$.

[7] The commitment scheme presented in Section 5 satisfies all needed requirements, but to simplify the presentation we use a generic description.

- $\texttt{KeyGen}_S(par)$: Pick $\alpha \leftarrow \mathbb{Z}_p$, set $g_1 \leftarrow g^\alpha$, and pick $g_2, h \leftarrow \mathbb{G}$. Furthermore, pick $u_0, \ldots, u_n \leftarrow \mathbb{Z}_p$, set $U_i \leftarrow g^{u_i}$, and define $F(m) = U_0 \prod_{i=1}^{n} U_i^{m_i}$ where m_i is the ith bit of m[8]. Finally pick a hash key $\mathtt{k} \in \mathcal{K}$ and return $pk_S = (\mathtt{k}, g_1, g_2, h, U_0, \ldots, U_n)$ and $sk_S = (g_2^\alpha, u_0, \ldots, u_n)$.
- $\texttt{KeyGen}_C(par)$: Pick $x, y \leftarrow \mathbb{Z}_p$ and set $u \leftarrow g^{x^{-1}}$ and $v \leftarrow g^{y^{-1}}$. Return $pk_C = (u, v)$ and $sk_C = (x, y)$.
- $\texttt{KeyGen}_V(par)$: Compute $(par, td) \leftarrow \mathcal{T}.\mathtt{G}(1^n)$ and return $(pk_V, sk_V) = (par, td)$.
- $\texttt{ConfirmedSign}$: Firstly, given input $(par, pk_S, sk_S, pk_C, m)$, where $pk_C = (u, v)$ and $sk_S = (g_2^\alpha, u_0, \ldots, u_n)$, the signer constructs a signature by picking $a, b, s \leftarrow \mathbb{Z}_p$, computing $t \leftarrow H_{\mathtt{k}}(pk_C||u^a||v^b||m)$ and $M = g^t h^s$, and setting $\sigma = (u^a, v^b, g_2^\alpha F(M)^{a+b}, s)$. The signature σ is then sent to the verifier.
  The signer and verifier then interact in the following protocol with $(par, pk_S, pk_C, pk_V, m, \sigma)$ as common input and the random choices (a, b) as private input to the signer.

$$\text{DV-}\Sigma\{(a,b) : u^a = \sigma_1 \wedge v^b = \sigma_2 \wedge e(g, F(M))^{a+b} = e(g, \sigma_3)/e(g_1, g_2)\}$$

  where M is computed as in the above.
- $\texttt{Extract}(par, sk_C, m, \sigma)$: Let $\sigma = (\sigma_1, \sigma_2, \sigma_3, s)$ and $sk_C = (x, y)$. Return the extracted signature $\sigma' = (pk_C, \sigma_1, \sigma_2, \sigma_1^x \sigma_2^y, \sigma_3, s)$.
- $\texttt{Verify}(par, pk_S, m, \sigma')$: Let $pk_S = (\mathtt{k}, g_1, g_2, h, U_0, \ldots, U_n)$ and $\sigma' = (pk_C, \sigma_1, \sigma_2, \sigma_3', \sigma_3, s)$, and return $\texttt{accept}$ if $e(g, \sigma_3) = e(g_1, g_2) e(\sigma_3', F(M))$ where $M = g^t h^s$ and $t = H_{\mathtt{k}}(pk_C||\sigma_1||\sigma_2||m)$.
- $\texttt{Confirm}_C$: Given $(par, pk_S, pk_C, pk_V, m, \sigma)$ as common input, where $pk_S = (\mathtt{k}, g_1, g_2, h, U_0, \ldots, U_n)$, the confirmer uses private input $sk_C = (x, y)$ and interacts with the verifier in the protocol

$$\text{DV-}\Sigma\{(x,y) : u^x = g \wedge v^y = g \wedge e(\sigma_1, F(M))^x e(\sigma_2, F(M))^y = e(g, \sigma_3)/e(g_1, g_2)\}$$

  where $M = g^t h^s$ and $t = H_{\mathtt{k}}(pk_C||\sigma_1||\sigma_2||m)$.
- $\texttt{Disavow}_C$: Given common and private input as in $\texttt{Confirm}_C$, the confirmer interacts with the verifier in the protocol

$$\text{DV-}\Sigma\{(x,y) : u^x = g \wedge v^y = g \wedge e(\sigma_1, F(M))^x e(\sigma_2, F(M))^y \neq e(g, \sigma_3)/e(g_1, g_2)\}$$

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

The following theorem reduces the unforgeability against passive adversaries of the above scheme to the unforgeability of Waters' signature scheme [39]. We refer the reader to [39] for a full description of this well known signature scheme.

Theorem 9 *Assume that the Waters' signature scheme [39] is unforgeable, $\mathcal{H}$ is a collision resistant hash function family, and the discrete logarithm problem is computationally hard in $\mathbb{G}$. Then the above on-line non-transferable signature scheme $\mathcal{S}$ is unforgeable against passive adversaries.*

Note that in [39], Waters' signatures are proved unforgeable assuming the computational Diffie-Hellman problem is hard, and that collision resistant hash functions can be constructed using this assumption as well [14]. The proof of the above theorem follows a similar strategy to [3], and will be given in the full version of the paper.

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

Theorem 10 *Assume that the above on-line non-transferable signature scheme $\mathcal{S}$ is unforgeable against passive adversaries, and that the decisional linear problem is computationally hard in $\mathbb{G}$. Then $\mathcal{S}$ is invisible against passive adversaries.*

The proof of the above theorem can be found in Appendix C.

From the above description, it is easy to see that the proof systems used to implement the confirm and disavow protocols are constructed as described in Section 5 and that the underlying sigma protocols satisfy the given requirements. However, before Theorem 4 can be applied, we need to verify that the scheme provides extended basic key security for confirmers. To see this, note that given the private signer key $sk = (g_2^\alpha, u_0, \ldots, u_n)$, only the discrete logarithm $x = \log_u g$ of the private confirmer key is needed to check validity of a signature[9], but an adversary recovering the full private confirmer key will have to compute $y = \log_v g$ as well. Hence, it is easy to reduce such an adversary to an algorithm computing discrete logarithms in $\mathbb{G}$. Furthermore, since knowledge of sk_S is sufficient to compute an extracted signature from a valid ordinary signature[10], the requirements for Theorem 6 are satisfied as well. Hence, Theorems 3, 4, 5, 6, 7 and 8 are all applicable, and we conclude that the above scheme is a secure on-line non-transferable signature scheme.

7 Conclusion

We have presented a new approach to the construction of on-line non-transferable signature schemes based on an extension of designated verifier proofs. Furthermore, we have proposed a concrete scheme that is provably secure in the standard model, is more efficient and meets a more natural and higher level of security than the proposal by Liskov and Micali.

8 Acknowledgement

The authors would like to thank Kenny Paterson for insightful comments and helpful suggestions.

References

1. Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Universal designated verifier signature proof (or how to efficiently prove knowledge of a signature). In Roy [34], pages 644–661.
2. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.
3. Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In *Public Key Cryptography*, volume 3958 of *LNCS*, pages 229–240. Springer, 2006.
4. Emmanuel Bresson, Dario Catalano, and Rosario Gennaro. Improved on-line/off-line threshold signatures. In *Public Key Cryptography*, volume 4450 of *LNCS*, pages 217–232. Springer, 2007.
5. Jan Camenisch and Markus Michels. Confirmer signature schemes secure against adaptive adversaries. In *EUROCRYPT*, pages 243–258. Springer, 2000.
6. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.

[9] Given m and $(\sigma_1, \sigma_2, \sigma_3, s)$, check that $e(\sigma_1^x, v)e(g, \sigma_2) = e((\sigma_3/g_2^\alpha)^{(u_0+\sum_i u_i M_i)^{-1}}, v)$ where $M = g^t h^s$ and $t = H_{\mathrm{k}}(pk_C||\sigma_1||\sigma_2||m)$.

[10] More specifically, `SimExtract` is constructed as follows: Given $sk_S = (g_2^\alpha, u_0, \ldots, u_n)$, m and a *valid* $(\sigma_1, \sigma_2, \sigma_3, s)$, compute $\sigma_3' = (\sigma_3/g_2^\alpha)^{(u_0+\sum_i u_i M_i)^{-1}}$ where $M = g^t h^s$ and $t = H_{\mathrm{k}}(pk_C||\sigma_1||\sigma_2||m)$, and return $\sigma' = (\sigma_1, \sigma_2, \sigma_3', \sigma_3, s)$.

7. 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.
8. Dario Catalano, Mario Di Raimondo, Dario Fiore, and Rosario Gennaro. Off-line/on-line signatures: Theoretical aspects and experimental results. In *Public Key Cryptography*, volume 4939 of *LNCS*, pages 101–120. Springer, 2008.
9. David Chaum. Zero-knowledge undeniable signatures. In *EUROCRYPT*, pages 458–464. Springer, 1990.
10. David Chaum. Designated confirmer signatures. In *EUROCRYPT*, pages 86–91. Springer, 1994.
11. David Chaum and Hans Van Antwerpen. Undeniable signatures. In *CRYPTO*, volume 435 of *LNCS*, pages 212–216. Springer, 1989.
12. Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, Proceedings*, volume 3494 of *LNCS*. Springer, 2005.
13. Ronald Cramer, editor. *Public Key Cryptography - PKC 2008, Proceedings*, volume 4939 of *LNCS*. Springer, 2008.
14. I. Damgård. Collision free hash functions and public key signature schemes. In *EUROCRYPT*, pages 203–216. Springer, 1987.
15. Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430. Springer, 2000.
16. Steven D. Galbraith and Wenbo Mao. Invisibility and anonymity of undeniable and confirmer signatures. In *CT-RSA*, volume 2612 of *LNCS*, pages 80–97. Springer, 2003.
17. Steven D. Galbraith, Wenbo Mao, and Kenneth G. Paterson. RSA-based undeniable signatures for general moduli. In *CT-RSA*, volume 2271 of *LNCS*, pages 200–217. Springer, 2002.
18. Craig Gentry, David Molnar, and Zulfikar Ramzan. Efficient designated confirmer signatures without random oracles or general zero-knowledge proofs. In Roy [34], pages 662–681.
19. Shafi Goldwasser and Erez Waisbard. Transformation of digital signature schemes into designated confirmer signature schemes. In *TCC*, volume 2951 of *LNCS*, pages 77–100. Springer, 2004.
20. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007/155, 2007. `http://eprint.iacr.org/`.
21. Xinyi Huang, Yi Mu, Willy Susilo, and Wei Wu. Provably secure pairing-based convertible undeniable signature with short signature length. In *Pairing*, volume 4575 of *LNCS*, pages 367–391. Springer, 2007.
22. Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, pages 143–154. Springer, 1996.
23. Caroline Kudla and Kenneth G. Paterson. Non-interactive designated verifier proofs and undeniable signatures. In *IMA Int. Conf.*, volume 3796 of *LNCS*, pages 136–154. Springer, 2005.
24. Kaoru Kurosawa and Swee-Huay Heng. 3-move undeniable signature scheme. In Cramer [12], pages 181–197.
25. Fabien Laguillaumie, Benoît Libert, and Jean-Jacques Quisquater. Universal designated verifier signatures without random oracles or non-black box assumptions. In *SCN*, volume 4116 of *LNCS*, pages 63–77. Springer, 2006.
26. Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *SCN*, volume 3352 of *LNCS*, pages 105–119. Springer, 2004.
27. Benoît Libert and Jean-Jacques Quisquater. Identity based undeniable signatures. In *CT-RSA*, volume 2964 of *LNCS*, pages 112–125. Springer, 2004.
28. Moses Liskov and Silvio Micali. Online-untransferable signatures. In Cramer [13], pages 248–267.
29. Markus Michels and Markus Stadler. Generic constructions for secure and efficient confirmer signature schemes. In *EUROCRYPT*, pages 406–421, 1998.
30. Jean Monnerat and Serge Vaudenay. Short 2-move undeniable signatures. In *VIETCRYPT*, volume 4341 of *LNCS*, pages 19–36. Springer, 2006.
31. Tatsuaki Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In *CRYPTO*, volume 839 of *LNCS*, pages 61–74. Springer, 1994.
32. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.
33. Le Trieu Phong, Kaoru Kurosawa, and Wakaha Ogata. New rsa-based (selectively) convertible undeniable signature schemes. In *AFRICACRYPT*, volume 5580 of *LNCS*, pages 116–134. Springer, 2009.
34. Bimal Roy, editor. *Advances in Cryptology - ASIACRYPT 2005, Proceedings*, volume 3788 of *LNCS*. Springer, 2005.
35. Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal designated-verifier signatures. In *ASIACRYPT*, volume 2894 of *LNCS*, pages 523–542. Springer, 2003.

36. Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Efficient extension of standard schnorr/rsa signatures into universal designated-verifier signatures. In *Public Key Cryptography*, volume 2947 of *LNCS*, pages 86–100. Springer, 2004.
37. Isamu Teranishi, Takuro Oyama, and Wakaha Ogata. General conversion for obtaining strongly existentially unforgeable signatures. In *INDOCRYPT*, volume 4329 of *LNCS*, pages 191–205. Springer, 2006.
38. Guilin Wang, Joonsang Baek, Duncan S. Wong, and Feng Bao. On the generic and efficient constructions of secure designated confirmer signatures. In *Public Key Cryptography*, volume 4450 of *LNCS*, pages 43–60. Springer, 2007.
39. Brent Waters. Efficient identity-based encryption without random oracles. In Cramer [12], pages 114–127.
40. Douglas Wikström. Designated confirmer signatures revisited. In *TCC*, volume 4392 of *LNCS*, pages 342–361. Springer, 2007.

A Proof of Theorem 3

Proof. Since the proof for $\overline{\mathcal{P}}$ is almost identical to the proof for $\mathcal{P}$, only the latter is given here. Furthermore, to simplify notation, we assume that Σ' and Σ'' share the same challenge space.

The algorithm $\mathtt{P}^{sim}(PK, i_P, sk_V, e)$, where $PK = (pk_S, pk_C, pk_V)$, $i_P \in \{1, 2\}$ and $e = (m, \sigma)$, is constructed as follows:

1. Pick a random challenge c and set $(a, c, z) \leftarrow \mathtt{Sim}_{\Sigma'}((pk_S, pk_C, pk_V, e), c)$ if $i_P = 1$ and $(a, c, z) \leftarrow \mathtt{Sim}_{\Sigma''}((pk_S, pk_C, pk_V, e), c)$ if $i_P = 2$. Compute a commitment $(com, aux) \leftarrow \mathtt{TdComm}(pk_V)$, and send (a, com) as the first message.
2. Upon receiving a challenge c', compute $w' = c - c'$ and $r' = \mathtt{TdOpen}(aux, w', sk_V)$, and send (w', r', z) as a response.

It is easy to verify that the above algorithm will cause an honest verifier to accept.

We will now show that the *interaction* of a simulated proof is indistinguishable from the interaction of a real proof. This will play a key role in the following proofs.

Lemma 11 *Assume the commitment scheme $\mathcal{T}$ has a perfect trapdoor property, and that Σ' and Σ'' are special honest verifier zero-knowledge. Then the interactions of $\mathtt{P}$ and $\mathtt{P}^{sim}$ are perfectly indistinguishable.*

Proof. Consider the experiment $\mathtt{Exp}^{\text{int-sim}}_{\mathcal{P},\mathcal{A}}$ defined in Figure 1. Let $\mathtt{Exp}^{\text{int-sim}'}_{\mathcal{P},\mathcal{A}}$ be an experiment similar to $\mathtt{Exp}^{\text{int-sim}}_{\mathcal{P},\mathcal{A}}$ but with the change that the challenge element is generated honestly if $b = 1$ instead of being picked at random from the element space i.e. $e^* \leftarrow \mathcal{E}$ in line 16 is replaced with $e^* \overset{s,sk^i}{\leftarrow} L(pk^i_{P'}, pk^j_{P''})$. Furthermore, let $\mathtt{Exp}^{\text{int-sim}'\text{-b}}_{\mathcal{P},\mathcal{A}}$ be the experiment in which the challenge bit b is chosen. We will now show that the view of a distinguisher $\mathcal{A}$ in experiment $\mathtt{Exp}^{\text{int-sim}'\text{-0}}_{\mathcal{P},\mathcal{A}}$ is distributed identically to the view of $\mathcal{A}$ in experiment $\mathtt{Exp}^{\text{int-sim}'\text{-1}}_{\mathcal{P},\mathcal{A}}$. This is done by considering the following sequence of games.

Game$_0$ This game is identical to the experiment $\mathtt{Exp}^{\text{int-sim}'\text{-0}}_{\mathcal{A},\mathtt{P}}$.

Game$_1$ In this game, we change how the commitment in $\mathtt{P}$ is computed and opened when interacting with $\mathcal{A}$ in the proof of validity for the challenge element e^*. Firstly, a random value c^* from the challenge space of the sigma protocols Σ' and Σ'' is picked in the beginning of the experiment. Then, in the first message of $\mathtt{P}$, a commitment $(com', aux) \leftarrow \mathtt{TdComm}(pk^{k^*}_V)$ is sent to $\mathcal{A}$ instead of an honestly computed commitment. When $\mathcal{A}$ sends a challenge c', the trapdoor $sk^{k^*}_V$ is used to compute $r' \leftarrow \mathtt{TdOpen}(aux, c^* - c', sk^{k^*}_V)$, and the opening $(c^* - c', r')$ is sent to $\mathcal{A}$ in the last message. Since $c^* - c'$ is distributed uniformly

in the challenge space, and the commitment scheme provides a perfect trapdoor property, the values $(com', (c^* - c', r'))$ must be distributed identically to the corresponding values in $\mathbf{Game}_0$, and hence, the view of $\mathcal{A}$ must be distributed identically to the view in $\mathbf{Game}_0$.

$\mathbf{Game}_2$ In this game, we change how the messages of the underlying sigma protocols are computed when interacting with $\mathcal{A}$ in the proof of validity for the challenge element e^*. Before interacting with $\mathcal{A}$, the simulated transcript $(a, c^*, z) \leftarrow \mathtt{Sim}_{\Sigma'}((pk_S^{i^*}, pk_C^{j^*}, pk_V^{k^*}, e^*), c^*)$ if $i_P^* = 1$ or $(a, c^*, z) \leftarrow \mathtt{Sim}_{\Sigma''}((pk_S^{i^*}, pk_C^{j^*}, pk_V^{k^*}, e^*), c^*)$ if $i_P^* = 2$ is computed. Then the messages (a, z) is used in the interaction with $\mathcal{A}$ instead of honestly following Σ' or Σ''. Since both Σ' and Σ'' are assumed to be special honest verifier zero-knowledge, (a, z) are perfectly indistinguishable from messages in a real interaction using challenge message c^*, and the view of $\mathcal{A}$ must be distributed identically to the view in $\mathbf{Game}_1$. However, $\mathbf{Game}_2$ is identical to $\mathtt{Exp}^{\text{int-sim}'\text{-}1}_{\mathcal{A},\mathtt{P}}$, and since the above yields that the view of $\mathcal{A}$ is distributed identically in $\mathbf{Game}_0$ and $\mathbf{Game}_2$, we conclude that the interactions of $\mathtt{P}$ and $\mathtt{P}^{sim}$ are perfectly indistinguishable. □

With the above lemma, it becomes easy proving the DV proof system interaction simulatable assuming the core signature scheme is invisible against passive attacks, since, intuitively, an adversary can only win by distinguishing an element of $L(pk_S, pk_C)$ from a random element of $\mathcal{E}$.

We construct an invisibility adversary $\mathcal{B}_{inv}$ from an interaction simulatability adversary $\mathcal{A}$ as follows. Firstly, $\mathcal{B}_{inv}$ is given public parameters par and a list of public keys PK as a part of an invisibility experiment, and runs $\mathcal{A}$ with input (par, PK). While $\mathcal{A}$ is running, $\mathcal{B}_{inv}$ answers *Corrupt* queries by forwarding these to his own corrupt oracle. *EGen* queries (i, j, s) are answered by forwarding the submitted seed s and (i, j) to $\mathcal{B}_{inv}$'s own *Sign* oracle and returning the obtained signature/message pair (s, σ). *Prove* queries are answered as follows.

- Given input the indices i, j, k, a prover index i_P and an element $e = (m, \sigma)$, $\mathcal{B}_{inv}$ submits (i, j, m, σ) to his `Extract` oracle. If the oracle returns $\perp$, $\mathcal{B}_{inv}$ returns $\perp$ to $\mathcal{A}$. Otherwise, σ must be a valid signature on m under pk_S^i and pk_C^j. $\mathcal{B}_{inv}$ then obtains sk_V^k through his `Corrupt` oracle and interacts with $\mathcal{A}$ by running $\mathtt{P}^{sim}((pk_S^i, pk_C^j, pk_V^k), i_P, e, sk_V^k)$. As shown in the above lemma, this interaction is perfectly indistinguishable from running $\mathtt{P}$, and will not affect the success probability of $\mathcal{A}$ since $e = (s, \sigma)$ corresponds to a valid signature.

At some point $\mathcal{A}$ outputs a challenge seed s^* and indices i^*, j^*, k^*, and $\mathcal{B}_{inv}$ forwards (i^*, j^*, s^*) as his own challenge message. $\mathcal{B}_{inv}$ then obtains a challenge signature σ^*, forwards $e^* = (s^*, \sigma^*)$ as the challenge element to $\mathcal{A}$, and then interacts with $\mathcal{A}$ by running $\mathtt{P}^{sim}((pk_S^{i^*}, pk_C^{j^*}, pk_V^{k^*}), i_P^*, e^*, sk_V^{k^*})$. Note that if e^* corresponds to a valid signature, this is perfectly indistinguishable from running $\mathtt{P}((pk_S^{i^*}, pk_C^{j^*}, pk_V^{k^*}), i_P^*, e^*, sk)$ where $sk \leftarrow_{i_P^*} (sk_S^{i^*}, sk_C^{j^*})$ due to Lemma 11.

After having received e^*, $\mathcal{A}$ can ask additional queries which $\mathcal{B}_{inv}$ answers as above. When $\mathcal{A}$ terminates with output b, $\mathcal{B}_{inv}$ returns b.

From the above, it is clear that an adversary $\mathcal{A}$ with non-negligible success probability will lead to $\mathcal{B}_{inv}$ having non-negligible success probability. Hence, assuming the core signature scheme is invisible against passive adversaries, $\mathcal{P}$ must be interaction simulatable. □

B Proof of Theorem 4

Proof. Since the proof for $\overline{\mathcal{P}}$ is almost identical to the proof for $\mathcal{P}$, only the latter is given here.

Assume an adversary $\mathcal{A}$ that breaks the soundness of the proof system $\mathcal{P}$ exists. Using $\mathcal{A}$, we will show how to construct algorithms that either break the binding under selective trapdoor openings property of the commitment scheme, break the extended key security for confirmers, or forge a signature of the core signature scheme.

Firstly, we consider a simple simulator $\mathcal{B}$ that interacts with $\mathcal{A}$ in the soundness experiment. For a security parameter 1^n, $\mathcal{B}$ generates public parameters $par \leftarrow \mathtt{Setup}(1^k)$ and primary, secondary and verifier key pairs $(pk_S^i, sk_S^i) \leftarrow \mathtt{KeyGen}_S$, $(pk_C^j, sk_C^j) \leftarrow \mathtt{KeyGen}_C$ and $(pk_V^k), sk_V^k) \leftarrow \mathtt{KeyGen}_V$ for $1 \leq i, j, k \leq l$. Then $\mathcal{B}$ runs $\mathcal{A}$ with input par and $\{pk_S^i, pk_C^i, pk_V^i\}_{i,j,k=1\ldots l}$.

Since $\mathcal{B}$ knows all private keys, *Corrupt*, *EGen*, *Prove* and *Sim* queries can be answered as in the ordinary soundness experiment. Eventually, $\mathcal{A}$ outputs a challenge element $e^* = (m^*, \sigma^*)$, indices i^*, j^*, k^* and a prover index i_P^*. $\mathcal{B}$ then interacts with $\mathcal{A}$ by running the algorithm $\mathtt{V}((pk_P^{i^*}, pk_C^{j^*}, pk_V^{k^*}), i_P^*, e^*)$. Since $\mathcal{A}$ is a assumed to be a successful adversary, the resulting verifier output will be `accept` with non-negligible probability. $\mathcal{B}$ then rewinds $\mathcal{A}$, and using the same random tape, replays the interaction with $\mathcal{A}$ but provides $\mathcal{A}$ with a different challenge when running $\mathtt{V}((pk_P^{i^*}, pk_C^{j^*}, pk_V^{k^*}), i_P^*, e^*)$ to obtain two transcripts, $((a, com), c, (z, w, r))$ and $((a, com), c', (z', w', r'))$. It is not obvious that two accepting transcripts can be obtained like this. However, as shown in [15], this can be achieved with non-negligible probability assuming $\mathcal{A}$ succeeds with non-negligible probability. We refer the reader to [15] for the details of this observation.

Let $\mathtt{comp}_S$ and $\mathtt{comp}_C$ be the events that $\mathcal{A}$ compromises $pk_S^{i^*}$ and $pk_C^{j^*}$, respectively. We will now consider the following four scenarios:

$$w \neq w' \qquad\qquad w = w' \wedge i_P^* = 1 \wedge \neg\mathtt{comp}_S$$
$$w = w' \wedge (i_P^* = 1 \wedge \mathtt{comp}_S \vee i_P^* = 2 \wedge \mathtt{comp}_C) \qquad\qquad w = w' \wedge i_P^* = 2 \wedge \neg\mathtt{comp}_C$$

Note that the above scenarios cover the entire probability space, and that a successful adversary will have to be successful in one of these scenarios. In the following we will show variants $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ of $\mathcal{B}$ that will successfully break one of the security assumptions in each of the different scenarios.

Scenario $w \neq w'$. In this scenario, $\mathcal{B}_1$ runs a binding under selective trapdoor openings experiment for the trapdoor commitment scheme, and receives a set of trapdoor parameters par_{td} which includes a security parameter 1^n. $\mathcal{B}_1$ then generates signature parameters $par \leftarrow \mathtt{Setup}(1^k)$, selects a special index $k' \in \{1, \ldots, l\}$ and sets $pk_V^{k'} \leftarrow par_{td}$. Since $\mathtt{KeyGen}_V$ corresponds to running $\mathtt{G}(1^k)$ for the trapdoor commitment scheme, $pk_V^{k'}$ will be a valid verifier key. For $1 \leq i, j, k \leq l$ $k \neq k'$, $\mathcal{B}_1$ generates keys (pk_S^i, sk_S^i), (pk_C^j, sk_C^j) and (pk_V^k, sk_V^k) in a similar way to $\mathcal{B}$ and runs $\mathcal{A}$ with input par and $\{pk_S^i, pk_C^j, pk_V^k\}_{i,j,k=1\ldots l}$. Since $\mathcal{B}_1$ holds all key pairs (pk_S^i, sk_S^i) and (pk_C^j, sk_C^j), `EGen` and `Prove` queries made by $\mathcal{A}$ can be answered in a similar way to $\mathcal{B}$. *Corrupt* and *Sim* queries are answered as follows:

- *Corrupt* queries: If $\mathcal{A}$ submits the query (k, V) where $k = k'$, $\mathcal{B}_1$ aborts. Otherwise the relevant private key is returned in a similar way to $\mathcal{B}$.
- *Sim* queries: For input (i, j, k, i_P, e) where $k \neq k'$, $\mathcal{B}_1$ runs $\mathtt{P}^{sim}((pk_P^i, pk_C^j, pk_V^k), i_P, e, sk_V^k)$. Otherwise, $\mathcal{B}_1$ responds as follows:
  - Firstly, $\mathcal{B}_1$ queries its commitment oracle $\mathcal{O}_c$ and obtains a commitment com, randomly picks a challenge c, and then runs $(a, c, z) \leftarrow \mathtt{Sim}_{\Sigma'}((pk_S^i, pk_C^j, pk_V^k, e), c)$ if $i_P = 1$ and $(a, c, z) \leftarrow \mathtt{Sim}_{\Sigma'}((pk_S^i, pk_C^j, pk_V^k, e), c)$ if $i_P = 2$. $\mathcal{B}_1$ then returns (a, com) as the first message to $\mathcal{A}$.

- Upon receiving a challenge c', $\mathcal{B}_1$ sets $w = c - c'$ and submits the commitment *com* and the value w to his opening oracle $\mathcal{O}_o$, and receives randomness r. $\mathcal{B}_1$ then sends the response (w, r, z) to $\mathcal{A}$.

It is easy to verify that the above corresponds to a valid interaction with $\mathtt{P}^{sim}$, the only difference being $\mathcal{B}_1$ using his oracles to open the commitment in the correct way instead of using the private verifier key.

Eventually, $\mathcal{A}$ outputs a challenge element $e^* = (m^*, \sigma^*)$, indices i^*, j^*, k^* and a prover index i_P^*. If $k^* \neq k'$, $\mathcal{B}_1$ aborts. Otherwise, $\mathcal{B}_1$ interacts with $\mathcal{A}$ by running $\mathtt{V}((pk_S^{i^*}, pk_C^{j^*}, pk_V^{k^*}), i_P, e)$, rewinds and replays $\mathcal{A}$ with a different challenge to obtain two transcripts $((a, com), c, (z, w, r))$ and $((a, com), c', (z', w', r'))$ in a similar way to $\mathcal{B}$.

In this scenario, it is assumed that $w \neq w'$. Furthermore, since the commitment *com* is the same in both transcripts, this means that $((w, r), (w', r'))$ is a valid attack against the binding property of the commitment scheme, and $\mathcal{B}_1$ submits this in the binding experiment.

Note that a successful $\mathcal{A}$ is not allowed to compromise $sk_V^{k^*}$, and with probability $1/l$, $\mathcal{B}_1$ will choose $k' = k^*$. In this case, $\mathcal{B}_1$ will not abort, will provide $\mathcal{A}$ with a perfect simulation, and will break the binding property of the commitment scheme with non-negligible probability assuming $\mathcal{A}$ is successful with non-negligible probability.

Scenario $w = w' \wedge (i_P^* = 1 \wedge \mathtt{comp}_S \vee i_P^* = 2 \wedge \mathtt{comp}_C)$. In this scenario, we will show that it is not possible for $\mathcal{A}$ to be successful. Consider the unmodified algorithm $\mathcal{B}$ given above which obtains two transcripts $((a, com), c, (z, w, r))$ and $((a, com), c', (z', w', r'))$. Since we assume $\mathcal{A}$ will compromise the private key indicated by i_P^* (i.e. sk_S if $i_P^* = 1$ and sk_C if $i_P^* = 2$), we must have $e \notin L(pk_S^{i^*}, pk_C^{j^*})$ for $\mathcal{A}$ to be successful. However since $w = w'$ and $c \neq c'$, $(a, c + w, z)$ and $(a, c' + w', z')$ must be two transcripts with the same first message a and different challenges $c + w \neq c' + w'$ for one of the underlying sigma protocol Σ' and Σ''. Hence, $\mathcal{B}$ can extract a witness for $e \in L(pk_S^{i^*}, pk_C^{j^*})$, contradicting the assumption that $\mathcal{A}$ is successful.

Scenario $w = w' \wedge i_P^* = 1 \wedge \neg\mathtt{comp}_S$. In this scenario, $\mathcal{B}_2$ runs an unforgeability experiment, receives public parameters *par* and a set of public keys $\{pk_S^i, pk_C^j, pk_V^k\}_{i,j,k=1\ldots l}$, and forwards these as input to $\mathcal{A}$. $\mathcal{B}_2$ forwards *Corrupt* and *EGen* queries from $\mathcal{A}$ to his own *Corrupt* and *Sign* oracles, respectively, and returns the answers to $\mathcal{A}$. If $\mathcal{A}$ makes a *Prove* query (i, j, k, i_P, e), $\mathcal{B}_2$ firstly obtains sk_C^j by submitting (j, C) to his `Corrupt` oracle, checks that $e \in L(pk_S^i, pk_C^j)$, and returns $\perp$ to $\mathcal{A}$ if this is not the case (note that the validity of a signature $e = (m, \sigma)$ can be checked with the private confirmer key). If $i_P = 2$, $\mathcal{B}_2$ interacts with $\mathcal{A}$ by running $\mathtt{P}((pk_S^i, pk_C^j, pk_V^k), i_P, e, sk_C^j)$. If $i_P = 1$, $\mathcal{B}_2$ responds as follows

– Firstly, $\mathcal{B}_2$ obtains sk_V^k by submitting (k, V) to his *Corrupt* oracle, and then interacts with $\mathcal{A}$ by running $\mathtt{P}^{sim}((pk_P^i, pk_C^j, pk_V^k), i_P, e, sk_V^k)$. Note that since e is valid, this is perfectly indistinguishable from running $\mathtt{P}((pk_S^i, pk_C^j, pk_V^k), i_P, e, sk_S^i)$ due to Lemma 11, and will not affect the success probability of $\mathcal{A}$.

Lastly, if $\mathcal{A}$ makes a *Sim* query (i, j, k, i_P, e), $\mathcal{B}_2$ obtains sk_V^k by submitting (k, V) to his *Corrupt* oracle, and interacts with $\mathcal{A}$ by running $\mathtt{P}^{sim}((pk_P^i, pk_C^j, pk_V^k), i_P, e, sk_V^k)$.

Eventually, $\mathcal{A}$ outputs a challenge element $e^* = (m^*, \sigma^*)$, indices i^*, j^*, k^* and a prover index i_P^*. $\mathcal{B}_2$ interacts with $\mathcal{A}$ by running $\mathtt{V}((pk_S^{i^*}, pk_C^{j^*}, pk_V^{k^*}), i_P^*, e)$, rewinds and replays $\mathcal{A}$ with a

different challenge to obtain two transcripts $((a, com), c, (z, w, r))$ and $((a, com), c', (z', w', r'))$, where $c \neq c'$, in a similar way to $\mathcal{B}$.

Since we in this scenario assume that $w = w'$ and that $i_P^* = 1$, $(a, c+w, z)$ and $(a, c'+w', z')$ must be two transcripts with the same first message a and different challenges $c+w \neq c'+w'$ for the underlying sigma protocol Σ'. Hence, $\mathcal{B}_2$ can extract the private prover input $x \leftarrow \mathtt{WExt}_{\Sigma'}((a, c+w, z), (a, c'+w', z'))$. Since it is assumed that (x, e) enables selective forgery of the signature scheme, $\mathcal{B}_2$ selects a m not submitted in a sign query, constructs a signature σ^* using (x, e), and submits σ^* in the unforgeability experiment.

$\mathcal{B}_2$'s simulation for $\mathcal{A}$ is perfect, and since we in this scenario assume that the event $\mathtt{comp}_S$ does not happen (i.e. $\mathcal{A}$ does not compromise the challenge signer key), $\mathcal{B}_2$ will be successful in the unforgeability experiment if $\mathcal{A}$ is successful in the soundness experiment.

Scenario $w = w' \wedge i_P^* = 2 \wedge \neg\mathtt{comp}_C$. In this scenario, $\mathcal{B}_3$ runs a key recovery experiment, and receives public parameters *par* and a public confirmer key pk_C. Then $\mathcal{B}_3$ picks a special index $j' \in \{1, \ldots, l\}$, sets $pk_C^{j'} \leftarrow pk_C$, and computes $(pk_S^i, sk_S^i) \leftarrow \mathtt{KeyGen}_S$, $(pk_C^j, sk_C^j) \leftarrow \mathtt{KeyGen}_C$ and $(pk_V^k, sk_V^k) \leftarrow \mathtt{KeyGen}_V$ for $1 \leq i, j, k \leq l$ $j \neq j'$. Lastly, $\mathcal{B}_3$ runs $\mathcal{A}$ with input *par* and $\{pk_S^i, pk_C^j, pk_V^k\}_{i,j,k=1,\ldots,l}$.

Since $\mathcal{B}_3$ knows all signer and verifier keys, *EGen* and *Sim* queries can be answered in a similar way to $\mathcal{B}$. *Corrupt* and *Prove* queries are answered as follows

- *Corrupt* queries: If $\mathcal{A}$ submits a query (j, C) where $j = j'$, $\mathcal{B}_3$ aborts. Otherwise, the relevant private key is returned to $\mathcal{A}$.
- *Prove* queries: If $\mathcal{A}$ submits a query (i, j, k, i_P, e) where $j = j'$, $\mathcal{B}_2$ submits (pk_S^i, sk_S^i, e) to his validity oracle. If e corresponds to an invalid signature, $\mathcal{B}_2$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}_2$ interacts with $\mathcal{A}$ by running $\mathtt{P}^{sim}((pk_P^i, pk_C^j, pk_V^k), i_P, e, sk_V^k)$. As argued above, since e is valid, this is perfectly indistinguishable from running $\mathtt{P}$ due to Lemma 11, and will not affect the success probability of $\mathcal{A}$. For all other queries having $j \neq j'$, $\mathcal{B}_2$ knows all relevant private keys and can answer the queries in a similar way to $\mathcal{B}$.

Eventually, $\mathcal{A}$ outputs a challenge element $e^* = (m^*, \sigma^*)$, indices i^*, j^*, k^* and a prover index i_P^*. If $j^* \neq j'$, $\mathcal{B}_2$ abort. Otherwise, $\mathcal{B}_2$ interacts with $\mathcal{A}$ in the $\mathtt{Prove}$ protocol by running $\mathtt{V}((pk_S^{i^*}, pk_C^{j^*}, pk_V^{k^*}), i_P^*, e^*)$, rewinds and replays $\mathcal{A}$ with a different challenge to obtain two transcripts $((a, com), c, (z, w, r))$ and $((a, com), c', (z', w', r'))$, where $c \neq c'$, in a similar way to $\mathcal{B}$.

In this scenario we assume that $w = w'$ and that $i_P^* = 2$. Hence, $(a, c+w, z)$ and $(a, c'+w', z')$ must be two transcripts with the same first message a and different challenges $c+w \neq c'+w'$ for the underlying sigma protocol Σ'', and $\mathcal{B}_3$ can extract the private prover input, the private confirmer key $sk_C^{j^*} \leftarrow \mathtt{WExt}_{\Sigma''}((a, c+w, z), (a, c'+w', z'))$, and returns $sk_C^{j^*}$ in the key recovery experiment.

Note that if $\mathcal{B}_2$ chooses $j' = j^*$ (which will happen with probability $1/l$), $\mathcal{B}_2$'s simulation will be perfect and $\mathcal{B}_2$ will recover $sk_C^{j^*}$ whenever $\mathcal{A}$ is successful in the soundness experiment. □

C Proof of Theorem 10

Proof. Since the verifier keys are not relevant when considering passive adversaries, we will ignore these in the following proof for simplicity.

We assume that an adversary $\mathcal{A}$ breaking the invisibility of the scheme exists. Let $\texttt{forge}$ be the event that $\mathcal{A}$ submits an extract query (i, j, m, σ) where pk_S^i is uncorrupted, and σ is a valid signature on m which was not obtained through a sign query (i, j, m). In the following we will construct algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ which will break the unforgeability of the scheme and the linear assumption in the events $\texttt{forge}$ and $\neg\texttt{forge}$, respectively.

Firstly assume that the event $\texttt{forge}$ happens. $\mathcal{B}_1$ runs an unforgeability experiment, receives public parameters par and a list of public keys $\{pk_S^i, pk_C^j\}_{i,j=1,\ldots,l}$, and forward these as input to $\mathcal{A}$. While running, $\mathcal{A}$ can ask *Corrupt*, *Sign* and *Extract* queries. $\mathcal{B}_1$ forwards *Corrupt* and *Sign* queries to his own corresponding oracles, and returns the answers to $\mathcal{A}$. If $\mathcal{A}$ makes an *Extract* query (i, j, m, σ), $\mathcal{B}_1$ submits (j, C) to his *Corrupt* oracle to obtain sk_C^j (note that $\mathcal{B}_1$ can corrupt any confirmer in the unforgeability experiment), checks if (i, j, m, σ) is a valid signature using sk_C^j, and returns $\perp$ to $\mathcal{A}$ if this is not the case. If (i, j, m, σ) is valid, $\mathcal{B}_1$ checks if pk_S^i is uncorrupted and if σ was not returned as a response to a sign query (i, j, m). If both conditions hold, $\mathcal{B}_1$ outputs (i, j, m, σ) and halts. Otherwise, $\mathcal{B}_1$ returns $\sigma' \leftarrow \texttt{Extract}(pk_S^i, sk_C^j, m, \sigma)$ to $\mathcal{A}$.

At some point, $\mathcal{A}$ outputs a challenge (i^*, j^*, m^*). As in the invisibility experiment, $\mathcal{B}_1$ flips a random coin $b \leftarrow \{0, 1\}$ and returns a random $\sigma^* \leftarrow \mathcal{S}_\sigma$ if $b = 0$. Otherwise, $\mathcal{B}_1$ returns σ^* obtained by submitting (i^*, j^*, m^*) to his *Sign* oracle. After receiving σ^*, $\mathcal{A}$ can ask additional *Corrupt*, *Sign* and *Extract* queries which $\mathcal{B}_1$ answers as above. If $\texttt{forge}$ happens, it is clear that $\mathcal{B}_1$ succeeds in winning in the unforgeability experiment.

Now assume that $\texttt{forge}$ does not happen. $\mathcal{B}_2$ will attempt to solve the decisional linear assumption i.e. $\mathcal{B}_2$ receives a description of a group $\mathbb{G}$ of order p and equipped with a pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, and elements $u, v, h, u^a, v^b, h^c \in \mathbb{G}$. $\mathcal{B}_2$'s goal is to decide if $c = a + b$. Firstly, $\mathcal{B}_2$ picks a hash family $\mathcal{H} = \{H_{\mathtt{k}} : \{0,1\}^* \rightarrow \mathbb{Z}_p\}$, an $r_g \leftarrow \mathbb{Z}_p$, and sets $g \leftarrow u^{r_g}$ and $par \leftarrow (\mathbb{G}, \mathbb{G}_T, e, p, g, \mathcal{H})$. Then $\mathcal{B}_2$ chooses a special signer index $i' \in \{1, \ldots, l\}$, picks $\mathtt{k}_{i'} \leftarrow \mathcal{K}$, $\alpha_{i'}, u_{i',0}, \ldots, u_{i',n} \leftarrow \mathbb{Z}_p$ and $g_{i',2}, h_{i'} \leftarrow \mathbb{G}$, and sets $pk_S^{i'} \leftarrow (\mathtt{k}_{i'}, g^{\alpha_{i'}}, g_{i',2}, h_{i'}, h^{u_{i',0}}, \ldots, h^{u_{i',n}})$ and $sk_S^{i'} = g_{i',2}^{\alpha_{i'}}$ (note that $sk_S^{i'}$ is only a partial private key, but is sufficient to run $\texttt{Sign}$). Furthermore, $\mathcal{B}_2$ picks a special confirmer index $j' \in \{1, \ldots, l\}$ and sets $pk_C^{j'} \leftarrow (u, v)$. For $1 \leq i \leq l$ $i \neq i'$ and $1 \leq j \leq l$ $j \neq j'$, $\mathcal{B}_2$ generates (pk_S^i, sk_S^i) and (pk_C^j, sk_C^j) using $\texttt{KeyGen}_S(par)$ and $\texttt{KeyGen}_C(par)$. Lastly $\mathcal{B}_2$ runs $\mathcal{A}$ with input $\{pk_S^i, pk_C^j\}_{i,j=1,\ldots,l}$.

While running, $\mathcal{A}$ can ask corrupt, sign and extract queries which are answered as follows.

- *Corrupt*: If $\mathcal{A}$ submits the query (i', S) or (j', C), $\mathcal{B}_2$ will abort. Otherwise, $\mathcal{B}_2$ simply returns the relevant private key.
- *Sign*: In the kth query (i_k, j_k, m_k), $\mathcal{B}_2$ returns $\sigma_k \leftarrow \texttt{Sign}(sk_S^{i_k}, pk_C^{j_k}, m_k)$ but remember the random choices $a_k, b_k \leftarrow \mathbb{Z}_p$ and stores $(i_k, j_k, m_k, \sigma_k, a_k, b_k)$ (note that $sk_S^{i'}$ is sufficient to run $\texttt{Sign}$).
- *Extract*: If $\mathcal{A}$ submits the query (i', j', m, σ) with the special indices (i', j'), $\mathcal{B}_2$ attempts to find a k such that $(i', j', m, \sigma, *, *) = (i_k, j_k, m_k, \sigma_k, a_k, b_k)$. Since we assume that $\texttt{forge}$ does not happen, such k will exist if (i', j', m, σ) is valid. In this case, $\mathcal{B}_2$ retrieves $\sigma_k = (\sigma_{k,1}, \sigma_{k,2}, \sigma_{k,3}, s_k)$ and returns the extracted signature $\sigma' = (pk_C^{j'}, \sigma_{k,1}, \sigma_{k,2}, g^{a_k+b_k}, \sigma_{k,3}, s_k)$. Otherwise, $\mathcal{B}_2$ returns $\perp$. For queries (i, j', m, σ) where $i \neq i'$ and $\sigma = (\sigma_1, \sigma_2, \sigma_3, s)$, $\mathcal{B}_2$ computes $t \leftarrow H_{k_i}(pk_C^{j'}||\sigma_1||\sigma_2||m)$, $M \leftarrow g^t h_i^s$ and $u^{a'+b'} \leftarrow (\sigma_3/(g_{i,2}^{\alpha_i}))^{(u_{i,0}+\sum_{k=1}^n u_{i,k}M_k)^{-1}}$, where the elements $\mathtt{k}_i, h_i$ are from pk_S^i, the elements $g_{i,2}^{\alpha_i}, u_{i,0}, \ldots, u_{i,n}$ are from sk_S^i, and a' and b' are unknown to $\mathcal{B}_2$. Then $\mathcal{B}_2$ checks if $e(u^{a'+b'}/\sigma_1, v) = e(u, \sigma_2)$ and returns $\perp$ if this is not the case (note that if this equation holds, we must have $\sigma_1 = u^{a'}$, $\sigma_2 = v^{b'}$ and

$\sigma_2 = g_{i,2}^{\alpha_i} F(M)^{a'+b'}$ for some $a', b' \in \mathbb{Z}_p$). Otherwise, $\mathcal{B}_2$ returns the extracted signature $\sigma' = (pk_C^{j'}, \sigma_1, \sigma_2, (u^{a'+b'})^{r_g}, \sigma_3, s)$. Lastly, for queries (i, j, m, σ) where $j \neq j^*$, $\mathcal{B}_2$ simply returns $\sigma' \leftarrow \texttt{Extract}(sk_C^j, pk_S^i, m, \sigma)$.

At some stage, $\mathcal{A}$ outputs a challenge message (i^*, j^*, m^*). If $(i^*, j^*) \neq (i', j')$, $\mathcal{B}_2$ aborts. Otherwise, $\mathcal{B}_2$ constructs a challenge signature by picking $s \leftarrow \mathbb{Z}_p$ and computing $t \leftarrow H_{\mathtt{k}_{i'}}(pk_C^{j'}||u^a||v^b||m)$, $M^* \leftarrow g^t h_{i'}^s$ and $\sigma^* \leftarrow (u^a, v^b, g_{i',2}^{\alpha_{i'}} (h^c)^{u_{i',0}+\sum_{k=1}^n u_{i',k} M_i^*}, s)$, where (u^a, v^b, h^c) are the elements received in the decisional linear problem. Note that if c is random, then σ^* will be a random element in $\mathbb{G}^3 \times \mathbb{Z}_p$, whereas if $c = a + b$, σ^* will be a valid signature on m since $(h^c)^{u_{i',0}+\sum_{k=1}^n u_{i',k} M_k^*} = (h^{u_{i',0}+\sum_{k=1}^n u_{i',k} M_k^*})^{a+b} = (U_{i',0} \prod_{k=1}^n U_{i',k}^{M_k^*})^{a+b}$.

$\mathcal{B}_2$ returns σ^* to $\mathcal{A}$ who can then ask additional *Corrupt*, *Sign* and *Extract* queries, but is not allowed to query σ^* to the extraction 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 if $\mathcal{B}_2$ guesses the correct challenge indices (i', j'), and from the above construction of the challenge signature, it is clear that $\mathcal{B}_2$ will solve the decisional linear problem if $\mathcal{A}$ breaks the invisibility of the scheme. $\square$