

# Non-interactive Designated Verifier Proofs and Undeniable Signatures

Caroline Kudla\* and Kenneth G. Paterson

Information Security Group  
Royal Holloway, University of London, UK  
{c.j.kudla,kenny.paterson}@rhul.ac.uk

**Abstract.** Non-interactive designated verifier (NIDV) proofs were first introduced by Jakobsson *et al.* and have widely been used as confirmation and denial proofs for undeniable signature schemes. There appears to be no formal security modelling for NIDV undeniable signatures or for NIDV proofs in general. Indeed, recent work by Wang has shown the original NIDV undeniable signature scheme of Jakobsson *et al.* to be flawed. We argue that NIDV proofs may have applications outside of the context of undeniable signatures and are therefore of independent interest. We therefore present two security models, one for general NIDV proof systems, and one specifically for NIDV undeniable signatures.

We go on to repair the NIDV proofs of Jakobsson *et al.*, producing secure NIDV proofs suited to combination with Chaum's original undeniable signature scheme resulting in a secure and efficient concrete NIDV undeniable signature scheme.

## 1 Introduction

Undeniable signatures were first presented in 1989 by Chaum and van Antwerpen [7], and were designed to have the property that signatures could be freely distributed, but were not self-authenticating. In other words, signatures cannot be verified without the cooperation of the signer. However, any party wrongly accused of having produced the signature can deny having produced the signature. The true signer may prove his authorship of an undeniable signature by running a confirmation protocol with a verifier, and a falsely implicated signer may deny his involvement by running a denial protocol with a verifier. Obviously, only the true signer should be able to successfully complete a confirmation protocol. Moreover the true signer should be unable to successfully complete a denial protocol for any of his signatures. Therefore the true signer cannot deny having produced his signatures.

---

\* This author is funded by Hewlett-Packard Laboratories.

The confirmation and denial protocols for the undeniable signature scheme of [7] were made zero-knowledge in [5], and this goes some way to ensuring that the signer has control over who can verify an undeniable signature. However, as was pointed out in [10], even though the confirmation protocol is zero-knowledge, the signer may still not always be able to control who is able to verify the validity of a signature if a group of verifiers cooperate. The undeniable signature scheme in [7] is also vulnerable to a blackmailing attack [12]. Jakobsson et al. [13] provide a solution, called designation of verifiers, to ensure that only a specified verifier can confirm an undeniable signature.

Informally, a designated verifier (DV) proof is a proof of correctness of some “statement” that either the prover or some designated verifier could have produced. If the prover created the proof, then the “statement” is correct, however a designated verifier could simulate a valid proof without a correct “statement”. A secure DV proof should convince the designated verifier of the correctness of the “statement” since the designated verifier knows that he did not create the proof himself. But no other party will be convinced of the validity of the proof since the designated verifier could have created it. In the context of undeniable signatures, a DV proof can be used to convince (only) the designated verifier of the validity of the undeniable signature.

Although the authors of [13] did not give a formal definition of DV proofs, they provided concrete examples of such proofs. The construction of DV proofs in [13] used trapdoor commitment schemes [2]. It was also shown there that the DV proofs could be made non-interactive. Such proofs are called NIDV proofs. However an attack on the concrete scheme of [13] was recently discovered by Wang [23], whereby a cheating signer can create a “non-standard” undeniable signature which the signer can prove valid via the NIDV confirmation proof and later deny via the denial proof. Wang proposed two ways to repair the scheme of [13], but did not offer any proofs of security. In fact, prior to this work, no formal definitions for NIDV proofs or for their security have ever been proposed, nor has a security model for NIDV undeniable signatures ever been developed.

For normal undeniable signature schemes, unforgeability and invisibility (or anonymity) are usually considered to be the key notions of security. As for the security of confirmation and denial proofs, the literature suggests that most authors have been content to simply prove that they are zero-knowledge and sound. This may suffice for the case where zero-knowledge confirmation and denial proofs are used, but it is unclear

whether these notions of security are satisfactory for NIDV proofs. In fact we argue here that they are not.

To summarise, little work has been done on security for NIDV undeniable signatures, and the earliest scheme [13] is now known to be insecure.

## 1.1 Our Contribution

We present a formal definition for NIDV proof systems which is compatible with the concrete schemes of [13]. We then propose a model of security for NIDV proof systems which we believe are of independent interest.

We then present a formal definition for NIDV undeniable signature schemes as well as a security model which models the security of both the core signature scheme and the NIDV confirmation and denial proof systems with which it is composed. Essentially, two NIDV proof systems are required to construct an NIDV undeniable signature scheme. The NIDV proof systems are for complementary languages, one providing confirmation proofs and the other denial proofs for the core signature scheme.

Our work represents the first time that a formal security model for NIDV undeniable signatures has been developed. The model does not require the signature scheme to be randomized. It is also a multi-party model, reflecting the fact that designated verifier proofs naturally involve more than one party, and that a party may play different roles at different times.

We consider NIDV proofs and NIDV undeniable signatures separately. One reason for doing so is that, in any application, signatures may exist independently of proofs. For example, a prover may generate a signature as a commitment to a message but only later provide an NIDV proof of its correctness, or a prover may generate many proofs for different designated verifiers on the same signature. A second reason is that NIDV proofs may also be useful in contexts other than undeniable signatures. One possible application of NIDV proofs is to deniable proofs of knowledge, in particular, proofs of knowledge of a private key. For example, when registering a public key with certification authority  $C$ ,  $A$  could demonstrate knowledge of the appropriate private key by presenting  $C$  with an NIDV proof of knowledge of the private key of  $A$ .

We go on to repair the NIDV proofs of [13], producing secure NIDV proof systems suited to combination with the full domain hash variant of Chaum's undeniable signature scheme [5, 7]. The NIDV proofs we obtain are actually a little shorter than those in [13]. Our work confirms that one of the fixes proposed by Wang [23] does indeed repair the NIDV proofs of [13]. The result is a concrete and efficient NIDV undeniable signature

scheme. Our paper concludes with some open problems and ideas for further extensions of our work.

## 1.2 Related Notions and Work

In parallel work, Lipmaa *et al.* [15] examine the security properties of what they refer to as designated verifier signature (DVS) schemes. In common with other authors [14, 19], they ascribe the term DVS to [13] and describe the concrete NIDV undeniable signature scheme of [13] as a DVS. In fact, this terminology was *never* used in [13]. Examination of [15] reveals that a DVS effectively combines an undeniable signature and an NIDV proof of correctness for that signature into a single entity. DVS, then, are closely related to (two-party) ring signatures. In contrast, the authors of [13] did not explicitly define such an object, preferring to keep undeniable signatures and their proofs separate. Our formal definition of an NIDV undeniable signature scheme also maintains the separation of signatures and proofs, and so is closer in spirit to the informal definitions in [13]. We have given several reasons why this separation is appropriate in the preceding section. It is unfortunate that this confusion over nomenclature has arisen in the literature, and we hope our work can help to clarify the situation.

Lipmaa *et al.* [15] go on to formalize and extend the attack of Wang [23] on [13] to DVS. They then propose two new security properties which are required for secure DVS, namely non-delegatability and disavowability. Although these notions are presented in the context of DVS, they could also be applied in the context of NIDV undeniable signatures. Our security model for NIDV undeniable signatures does in fact capture the notion of non-delegatability, but we do not consider disavowability to be a feature of NIDV proofs, and our concrete examples do not have this property since they are unconditionally non-transferable.

Another related notion is that of strong designated verifier (SDV) proofs [13, 19, 22], which provide stronger security guarantees than NIDV proofs. SDV proofs provide similar properties to NIDV proofs except that only the designated verifier is able to verify the proofs produced, since the verification algorithm requires the private key of the designated verifier. By contrast, NIDV proofs are universally verifiable, but only convincing to the designated verifier.

Also related to NIDV proofs (or more specifically to NIDV undeniable signatures) are universal designated verifier (UDV) signatures [14, 20, 21]. Although at first glance these appear to be similar to NIDV undeniable

signatures, they are quite different. UDV signature schemes produce signatures which are universally verifiable. However any party in possession of a valid UDV signature on message  $M$  from signer  $S$  can provide (to any verifier of their choice) a designated verifier proof that they possess a valid UDV signature on  $M$  by  $S$ . On the other hand, NIDV undeniable signatures are not universally verifiable, and only the signer is able to produce designated verifier proofs of a signature's validity. We do not consider UDV signatures in this paper.

## 2 Preliminaries

Let  $G$  be a finite, multiplicative group of prime order  $q$ , and let  $g$  be a generator of  $G$ . We denote by  $DL(g, h)$  the discrete logarithm of  $h$  with respect to base  $g$  in group  $G$ . So  $g^{DL(g, h)} = h$  in  $G$ . We also informally define the following problems in  $G$ :

**Discrete Logarithm (DL) Problem:** Given  $g, g^a \in G$ ,  $a \in_R \mathbb{Z}_q$ , compute  $a$ .

**Computational Diffie-Hellman (CDH) Problem:** Given  $g, g^a, g^b \in G$ ,  $a, b \in_R \mathbb{Z}_q$ , compute  $g^{ab}$ .

**Decisional Diffie-Hellman (DDH) Problem:** Given  $g, g^a, g^b, g^c \in G$ ,  $a, b \in_R \mathbb{Z}_q$ , determine whether  $c = ab \bmod q$ .

## 3 Non-interactive Designated Verifier Proof Systems

We now present a formal definition for NIDV proof systems. This formal definition was previously lacking in [13] but our definitions are compatible with the concrete scheme of [13].

A non-interactive designated verifier (NIDV) proof system is defined with respect to some family of languages  $\mathcal{L}$ . The goal of an NIDV proof system is to prove the membership of elements  $e$  in a language  $L \in \mathcal{L}$ . An NIDV proof system consists of the following algorithms:

- A probabilistic *Setup* algorithm which takes a security parameter  $l$  as input and returns the system parameters  $params$  and a description of a family of languages  $\mathcal{L}$ . Amongst the public parameters  $params$  are descriptions of the following spaces: a public key space  $\mathcal{PK}$ , a private key space  $\mathcal{SK}$ , an element space  $\mathcal{E}$  and a proof space  $\mathcal{P}$ .
- A probabilistic *KeyGen* algorithm which takes as input the public parameters  $params$  and returns a key pair  $(x, X)$  where  $x \in \mathcal{SK}$  is a private key and  $X \in \mathcal{PK}$  is the corresponding public key.

- A proof generation algorithm  $PGen$  which takes as input  $\langle x_P, X_V, e \rangle$  where  $x_P \in \mathcal{SK}$ ,  $X_V \in \mathcal{PK}$ , and  $e \in \mathcal{E}$  with  $e \in L(X_P)$ , and produces an NIDV proof  $\pi \in \mathcal{P}$  for  $e$ .
- A verification algorithm  $PVerify$  which takes as input  $\langle X_P, X_V, e, \pi \rangle$  where  $X_P, X_V \in \mathcal{PK}$ ,  $e \in \mathcal{E}$  and  $\pi \in \mathcal{P}$ , and outputs Accept or Reject.

Note that we parameterize the languages  $L \in \mathcal{L}$  by public keys, and for any public key  $X \in \mathcal{PK}$ ,  $L(X) \in (\mathcal{E})$ . This parametrization will be needed for the proof systems required for use with undeniable signatures but is not necessary in general.

## 4 Security of NIDV Proof Systems

We say that an NIDV proof system is secure if it satisfies the notions of correctness, non-transferability and soundness. These are defined next.

### 4.1 Correctness

An NIDV proof system is *correct* if when  $PGen$  is run on any input  $x_P \in \mathcal{SK}$ ,  $X_V \in \mathcal{PK}$ ,  $e \in \mathcal{E}$  and outputs some  $\pi \in \mathcal{P}$ , then  $PVerify$  on input  $\langle X_P, X_V, e, \pi \rangle$  outputs Accept.

### 4.2 Non-transferability

We say that an NIDV proof system is *non-transferable* if there exists a polynomial time algorithm  $A$  that on input tuples  $\langle X_P, x_V, e \rangle$ , where  $X_P \in \mathcal{PK}$ ,  $x_V \in \mathcal{SK}$ ,  $e \in \mathcal{E}$ , but where  $e$  is not necessarily in  $L(X_P)$ , produces proofs  $\pi \in \mathcal{P}$  such that  $\langle X_P, X_V, e, \pi \rangle$  is accepted by  $PVerify$  and the distribution of proof  $\pi$  is polynomially indistinguishable from proof  $\pi'$  produced by  $PGen$  when run on inputs  $\langle x_P, X_V, e' \rangle$  where  $e' \in \mathcal{E}$  and  $e' \in L(X_P)$ .

### 4.3 Soundness

Soundness of an NIDV proof system is defined via the following game between a challenger  $C$  and an adversary  $E$ :

**Setup:**  $C$  runs Setup and KeyGen for a given security parameter  $l$  to obtain the public parameters  $params$ , a description of a family of languages  $\mathcal{L}$ , as well as a set of public and private key pairs  $(X_i, x_i)$ .  $C$  sets up each participant oracle  $I$  with its public and private keys

$X_I, x_I$ .  $E$  is given  $params, \mathcal{L}$  and the public keys  $\{X_i\}$  of all participants while  $C$  retains the private keys  $\{x_i\}$ . We define the set of all participants' public keys to be  $\mathcal{X}$ .

$E$  can make the following types of query to the participant oracles:

**EGen Queries:**  $E$  can make an EGen query to an oracle  $P$  with public key  $X_P$  (possibly with input some  $seed$ ). The oracle outputs an element  $e \in L(X_P)$ .

**PGen Queries:**  $E$  can make a PGen query to oracle  $P$  with public key  $X_P$  on input  $\langle X_V, e \rangle$  where  $e \in \mathcal{E}$ ,  $X_V \in \mathcal{X}$ . If  $e \in L(X_P)$ , the oracle produces an NIDV proof  $\pi \in \mathcal{P}$  from  $P$  to  $V$  for  $e$ . If  $e \notin L(X_P)$  then the oracle outputs “invalid”.

**FakePGen Queries:**  $E$  can make a FakePGen query to oracle  $V$  with public key  $X_V$  on input  $\langle X_P, e \rangle$  where  $e \in \mathcal{E}$ ,  $X_P \in \mathcal{X}$ . The oracle runs algorithm  $A$  to produce an NIDV proof  $\pi \in \mathcal{P}$  for  $e$ . The oracle outputs  $\pi$ .

**Corrupt Queries:**  $E$  can request the private key  $x_I$  of any oracle  $I$ .

**Output:** Finally  $E$  outputs  $\langle X_P^*, X_V^*, e^*, \pi^* \rangle$ , where  $X_P^*, X_V^* \in \mathcal{X}$  and  $X_V^*$  is uncorrupted,  $e^* \in \mathcal{E}$  and  $\pi^* \in \mathcal{P}$ .  $E$  wins if  $\langle X_P^*, X_V^*, e^*, \pi^* \rangle$  is accepted by PVerify,  $\pi^*$  was not the output of some PGen query to  $P^*$  on  $\langle X_V^*, e^* \rangle$  or some FakePGen query to  $V^*$  on  $\langle X_P^*, e^* \rangle$ , and either:

1.  $X_P^*$  is uncorrupted, or
2.  $e^* \notin L(X_P^*)$ .

**Definition 1.** We say that an NIDV proof system is sound if the probability of success of any polynomially bounded adversary in the above game is negligible (as a function of the security parameter  $l$ ).

#### 4.4 Notes on the Security Definitions for NIDV Proof Systems

**Soundness** The soundness definition guarantees that if an uncorrupted verifier receives a valid NIDV proof, then it was created using the private key  $x_P$  and  $e \in L(X_P)$ . So a prover cannot cheat. Soundness also guarantees that no-one other than  $P$  can convince an uncorrupted designated verifier that  $e \in L(X_P)$ . In the context of undeniable signatures, this means that no-one other than the real signer is able to produce an NIDV proof of a valid undeniable signature that will be accepted by an uncorrupted designated verifier. This is essential for the security of undeniable signatures. The model for soundness is multiparty, reflecting the fact that NIDV proofs naturally involve more

than one party, and that a party may play different roles at different times.

**Non-transferability** The existence of algorithm  $A$  that can be run by  $V$  to create an NIDV proof for any element  $e$  (not necessarily in  $L$ ) ensures that no-one besides  $V$  will be convinced by an NIDV proof for  $e$ .

We note that we do not require the elements  $e$  and  $e'$  to be indistinguishable for non-transferability, rather only the proofs  $\pi$  and  $\pi'$  are required to be indistinguishable. If an outside party can already distinguish elements in  $L$  from elements not in  $L$ , then they have no need of NIDV proofs for  $L$ . However an NIDV proof for an element  $e$  should not give any extra information regarding  $e$  to parties other than the designated verifier.

**FakePGen queries** The existence of algorithm  $A$  from Section 4.2 also enables oracles to answer FakePGen queries. We consider it important to model such queries since an adversary may have access to such “faked” NIDV proofs that are produced by dishonest verifiers using algorithm  $A$ .

## 5 NIDV Undeniable Signature Schemes

As mentioned earlier, the main application of NIDV proofs has historically been in undeniable signatures, even though the current security models for undeniable signatures do not seem to support NIDV proofs. We now present a formal definition for NIDV undeniable signature schemes.

**Definition 2.** *An NIDV undeniable signature scheme consists of a core signature scheme as well as NIDV confirmation and denial proof systems. The core signature scheme consists of the following algorithms:*

- A probabilistic *Setup* algorithm which takes a security parameter  $l$  as input and returns the system parameters  $params$ . Amongst the public parameters are descriptions of the following spaces: a public key space  $\mathcal{PK}$ , a private key space  $\mathcal{SK}$ , a message space  $\mathcal{M}$  and a signature space  $\mathcal{S}$ .
- A probabilistic *KeyGen* algorithm which takes as input the public parameters  $params$  and returns a key pair  $(x, X)$  where  $x \in \mathcal{SK}$  and  $X \in \mathcal{PK}$ .
- A (possibly probabilistic) signature generation algorithm *Sign* which on input  $\langle x, m \rangle$  where  $x \in \mathcal{SK}, m \in \mathcal{M}$ , produces an undeniable signature  $\sigma \in \mathcal{S}$ .

The core signature scheme defines a language  $L(X)$  for each public key  $X$ , where  $L(X) = \{(m, \sigma) : \sigma = \text{Sign}(x, m)\}$ . In other words,  $L(X)$  is the language of all possible valid message and signature pairs for public key  $X$  and  $\overline{L(X)}$  is the language of all invalid message and signature pairs for public key  $X$ . The family of languages  $\mathcal{L}$  is defined as  $\mathcal{L} = \{L(X) : X \in \mathcal{PK}\}$  and  $\overline{\mathcal{L}}$  is defined as  $\overline{\mathcal{L}} = \{\overline{L(X)} : X \in \mathcal{PK}\}$ .

The families of languages  $\mathcal{L}$  and  $\overline{\mathcal{L}}$  parameterize the confirmation and denial proofs. The confirmation proof is an NIDV proof system  $\mathcal{C}$  for  $\mathcal{L}$ , and the denial proof is an NIDV proof system  $\mathcal{D}$  for  $\overline{\mathcal{L}}$ . The setup algorithms for  $\mathcal{C}$  and  $\mathcal{D}$  use the public key space  $\mathcal{PK}$ , the private key space  $\mathcal{SK}$ , and set the element space  $\mathcal{E}$  to be  $\mathcal{M} \times \mathcal{S}$ . The proof spaces for  $\mathcal{C}$  and  $\mathcal{D}$  are denoted  $\mathcal{P}_C$  and  $\mathcal{P}_D$  respectively. The following algorithms then make up the confirmation and denial proofs.

- A confirmation proof generation algorithm *ConfGen* which, on input  $\langle x_P, X_V, m, \sigma \rangle$  where  $x_P \in \mathcal{SK}$ ,  $X_V \in \mathcal{PK}$ ,  $(m, \sigma) \in L(X_P)$  runs *PGen* of  $\mathcal{C}$  on  $\langle x_P, X_V, (m, \sigma) \rangle$ .
- A confirmation proof verification algorithm *ConfVerify* which, on input  $\langle X_P, X_V, m, \sigma, \pi_C \rangle$  where  $X_P, X_V \in \mathcal{PK}$ ,  $(m, \sigma) \in \mathcal{E}$  and  $\pi_C \in \mathcal{P}_C$  runs *PVerify* of  $\mathcal{C}$  on  $\langle X_P, X_V, (m, \sigma), \pi_C \rangle$ .
- A denial proof generation algorithm *DenyGen* which, on input  $\langle x_P, X_V, m, \sigma \rangle$  where  $x_P \in \mathcal{SK}$ ,  $X_V \in \mathcal{PK}$ ,  $(m, \sigma) \in \overline{L(X_P)}$  runs *PGen* of  $\mathcal{D}$  on  $\langle x_P, X_V, (m, \sigma) \rangle$ .
- A denial proof verification algorithm *DenyVerify* which, on input  $\langle X_P, X_V, m, \sigma, \pi_D \rangle$  where  $X_P, X_V \in \mathcal{PK}$ ,  $(m, \sigma) \in \mathcal{E}$  and  $\pi_D \in \mathcal{P}_D$  runs *PVerify* of  $\mathcal{D}$  on  $\langle X_P, X_V, (m, \sigma), \pi_D \rangle$ .

## 6 Security of NIDV Undeniable Signatures

In analyzing the security of NIDV undeniable signatures, we consider the security of the confirmation and denial proofs being used, and their composition with the core signature scheme.

Unless explicitly stated, we will represent the public key of a participant  $I$  by  $X_I$ , and the private key as  $x_I$ .  $P$  will in general represent a prover, and  $V$  a verifier.

**Definition 3.** *The confirmation (denial) proof of an NIDV undeniable signature is secure if  $\mathcal{C}$  ( $\mathcal{D}$ ) is a secure NIDV proof system for  $\mathcal{L}$  ( $\overline{\mathcal{L}}$ ). That is,  $\mathcal{C}$  ( $\mathcal{D}$ ) is correct, non-transferable and sound.*

## 6.1 The Security of the Core Signature Scheme

The security of the core signature scheme is defined via the following notions:

**Unforgeability** Unforgeability of an undeniable signature scheme is defined via the following game between a challenger  $C$  and an adversary  $E$ :

**Setup:**  $C$  runs the Setup and KeyGen algorithms for a given security parameter  $l$  to obtain the public parameters  $params$  as well as a set of public and private key pairs  $(X_i, x_i)$ .  $E$  is given  $params$  and the set of public keys  $\{X_i\}$  of all participants while  $C$  retains the private keys  $\{x_i\}$ .  $C$  sets up each participant oracle  $I$  with its public and private keys  $X_I, x_I$ . We define the set of all participants' public keys to be  $\mathcal{X}$ .  $E$  can make the following types of query to the participant oracles:

**Sign Queries:**  $E$  can make a Sign query to any participant  $I$  with public key  $X_I$  on input  $m$  where  $m \in \mathcal{M}$ , and the oracle runs Sign on  $\langle x_I, m \rangle$  to produce a signature  $\sigma \in \mathcal{S}$ . The oracle outputs  $\sigma$ .

**Conf/Deny Queries:**  $E$  can make a Conf/Deny query to any participant  $P$  with public key  $X_P$  on input  $\langle X_V, m, \sigma \rangle$  where  $X_V \in \mathcal{X}$ ,  $m \in \mathcal{M}$  and  $\sigma \in \mathcal{S}$ . If  $(m, \sigma) \in L(X_P)$  then the oracle runs ConfGen on  $\langle x_P, X_V, m, \sigma \rangle$  to produce an NIDV proof  $\pi_C \in \mathcal{P}_C$  which it outputs, otherwise it runs DenyGen on  $\langle x_P, X_V, m, \sigma \rangle$  to produce an NIDV proof  $\pi_D \in \mathcal{P}_D$  which it outputs.

**FakeConf Queries:**  $E$  can make a FakeConf query to any participant  $V$  with public key  $X_V$  on input  $\langle X_P, m, \sigma \rangle$  where  $X_V \in \mathcal{X}$ ,  $m \in \mathcal{M}$  and  $\sigma \in \mathcal{S}$ . The oracle runs the algorithm  $A$  of the NIDV proof system  $C$  to produce an NIDV proof  $\pi_C \in \mathcal{P}_C$  which it outputs.

**FakeDeny Queries:**  $E$  can make a FakeDeny query to any participant  $V$  with public key  $X_V$  on input  $\langle X_P, m, \sigma \rangle$  where  $X_V \in \mathcal{X}$ ,  $m \in \mathcal{M}$  and  $\sigma \in \mathcal{S}$ . The oracle runs the algorithm  $A$  of the NIDV proof system  $D$  to produce an NIDV proof  $\pi_D \in \mathcal{P}_D$  which it outputs.

**Corrupt Queries:**  $E$  can make a corrupt query to any participant  $I$  with public key  $X_I$ , and the oracle outputs  $x_I$ .

**Output:** Finally  $E$  produces  $X_P^* \in \mathcal{X}$ ,  $m^* \in \mathcal{M}$  and  $\sigma^* \in \mathcal{S}$ , where  $X_P^*$  is uncorrupted and  $\sigma^*$  was not the output of some previous Sign query to  $P^*$  on  $m^*$ .  $E$  wins the game if  $(m^*, \sigma^*) \in L(X_P^*)$ .

**Definition 4.** We say that an undeniable signature scheme is unforgeable if the probability of success of any polynomially bounded adversary in the above game is negligible in  $l$ .

**Invisibility** Invisibility of an undeniable signature scheme is defined via the following game between a challenger  $C$  and an adversary  $E$ :

**Setup:** This is as in the Unforgeability game above.

**Phase 1:** The adversary can make Sign, Conf/Deny, FakeConf, FakeDeny and Corrupt queries, and these are all answered as in the Unforgeability game.

**Challenge:**  $E$  produces  $m^* \in \mathcal{M}$ ,  $X_P^* \in \mathcal{X}$ , where  $X_P^*$  is uncorrupted. In addition, if the Sign algorithm is deterministic, then  $E$  should not have previously made a Sign query to  $P^*$  on  $m^*$  in Phase 1.  $C$  chooses a random bit  $b$  and if  $b = 0$ ,  $C$  sets  $\sigma^* = r$  where  $r$  is randomly chosen from  $\mathcal{S}$ , otherwise  $C$  sets  $\sigma^* = \text{Sign}(x_P^*, m^*)$ .  $C$  gives  $\sigma^*$  to  $E$ .

**Phase 2:** Again  $E$  can make queries as in Phase 1, except that  $E$  cannot make a Conf/Deny query to  $P^*$  on  $\langle X_V, m^*, \sigma^* \rangle$  for any  $X_V$ . If the signature algorithm Sign is deterministic,  $E$  is also forbidden from making a Sign query to  $P^*$  on  $m^*$ .

**Output:** Finally  $E$  outputs a bit  $b'$  and wins the game if  $b' = b$ .

**Definition 5.** We say that an undeniable signature scheme is invisible if the probability of success of any polynomially bounded adversary in the above game is negligible in  $l$ .

## 6.2 Notes on the Security Definitions for Undeniable Signatures

**Correctness** Although we do not explicitly define correctness for NIDV undeniable signatures, correctness is handled by the correctness of proof systems  $\mathcal{C}$  and  $\mathcal{D}$ .

**Unforgeability** Our model of unforgeability differs from security models for normal undeniable signatures in two main ways. Firstly it is multiparty due to the multiparty nature of the NIDV confirmation and denial proofs. Secondly, we allow the adversary to make FakeConf and FakeDeny queries. We consider these to be necessary since an adversary may conceivably have access to such “fake” proofs produced by dishonest designated verifiers.

**Invisibility** We include the notion of invisibility in our model of security rather than anonymity. Anonymity, which could be defined in a similar way to [11], captures the notion that an adversary cannot determine which of two possible signers created a given signature. Analogous results to those in [11] could be used to show that invisibility is the stronger notion and implies anonymity. However we feel

that the stronger definition of invisibility is appropriate for NIDV undeniable signatures since we model the existence of fake NIDV proofs and their corresponding (possibly fake) signatures, and these should be indistinguishable from true signatures and NIDV proofs.

**Determinism** Our definitions of unforgeability and invisibility encompass both non-deterministic and deterministic undeniable signatures. We assume that signers can identify their own valid signatures. For deterministic undeniable signatures this is trivial because signers can just re-sign a message, but in the case of non-deterministic (or randomized) undeniable signatures this may be non-trivial.

We note that in the deterministic case, invisibility actually implies unforgeability, since an adversary who can forge signatures can trivially win the invisibility game. However for randomized signatures, these properties are distinct. We keep the properties distinct when proving the security of a deterministic undeniable signature scheme later in the paper because it makes the proofs easier.

## 7 A Concrete NIDV Undeniable Signature Scheme

We present the full domain hash variant of the undeniable signature scheme of Chaum [5] with NIDV confirmation and denial proofs.

### 7.1 The Core Signature

**Setup** For some security parameter  $l$ , let  $p$  and  $q$  be large primes, where  $q|(p-1)$ . Let  $G$  be a multiplicative subgroup of  $\mathbb{Z}_p^*$  of order  $q$  and let  $g$  be a generator of  $G$ . We also assume that  $H_1 : \{0,1\}^* \rightarrow G$  is a cryptographic hash function. For example, such a hash function may be constructed by using a standard hash function to map the input to a bitstring representing an integer, reducing that integer modulo  $p$  and then exponentiating the result to the power  $(p-1)/q$  modulo  $p$ . We set  $\mathcal{PK} = \mathcal{S} = G$ ,  $\mathcal{M} = \{0,1\}^*$  and  $\mathcal{SK} = \mathbb{Z}_q$ . The public parameters are  $params = (p, q, g, H_1, \mathcal{PK}, \mathcal{SK}, \mathcal{M}, \mathcal{S})$ .

**KeyGen** To set up a user  $I$ 's public and private keys, the private key  $x_I$  is chosen at random from  $\mathbb{Z}_q$ , and the public key is  $X_I = g^{x_I} \bmod p$ .

**Sign** On input  $\langle x_I, m \rangle$  where  $x_I \in \mathbb{Z}_q$ ,  $m \in \{0,1\}^*$ , compute  $\sigma = H_1(m)^{x_I} \bmod p$ . Output  $\sigma$ .

### 7.2 The Confirmation and Denial Proofs

The Sign algorithm defines a language  $L(X_I) = \{(m, \sigma) : \sigma = \text{Sign}(x_I, m)\}$  for each public key  $X_I$ . For the above signature scheme, we can write

$L(X_I) = \{(m, \sigma) : DL(\sigma, H_1(m)) = DL(x_I, g)\}$ . In other words,  $L(X_I)$  is the language of all possible message and signature pairs  $(m, \sigma)$  where the discrete logarithm of  $\sigma$  to the base  $H_1(m)$  equals the discrete logarithm of  $X_I$  to the base  $g$  modulo  $p$ . The family of languages  $\mathcal{L}$  is defined as  $\mathcal{L} = \{L(X_I) : X_I \in G\}$ .

We can now define confirmation and denial proofs with respect to the languages  $L(X_I)$ .

**Confirmation proof** The confirmation proof requires a secure NIDV proof system  $\mathcal{C}$  for  $\mathcal{L}$ . Informally,  $\mathcal{C}$  must prove the equality of two discrete logarithms (EDL).

**Denial proof** The denial proof requires a secure NIDV proof system  $\mathcal{D}$  for  $\overline{\mathcal{L}}$ . Informally,  $\mathcal{D}$  must prove the inequality of two discrete logarithms (IDL).

### 7.3 A Concrete NIDV EDL Proof System

The NIDV proof we present is a slight modification of the scheme of Jakobsson et al. [13] since the original proof was shown to be insecure by Wang [23].

Since our NIDV EDL proof will be used with the above undeniable signature scheme, the Setup algorithm will be identical to that in Section 7.1 except that in addition we require another cryptographic hash function  $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ , and we define the spaces  $\mathcal{E} = \mathcal{M} \times \mathcal{S}$  and  $\mathcal{P} = \mathbb{Z}_q^4$ . KeyGen will be exactly as in Section 7.1. The family of languages will be defined by the Sign algorithm of the concrete scheme as described above in Section 7.2. We still need to define the PGen and PVerify algorithms.

**NIDV EDL PGen** On input  $\langle x_P, X_V, m, \sigma \rangle$  where  $x_P \in \mathcal{SK}$ ,  $X_V \in \mathcal{PK}$ ,  $m \in \mathcal{M}$  and  $\sigma \in \mathcal{S}$ , the algorithm picks random  $w, r, t \in \mathbb{Z}_q$  and computes:

$$\begin{aligned} c &= g^w X_V^r \bmod p \\ G &= g^t \bmod p \\ M &= H_1(m)^t \bmod p \\ h &= H_2(c, G, M, m, \sigma, X_P) \\ d &= t - x_P(h + w) \bmod q \end{aligned}$$

The algorithm outputs  $\pi = \langle w, r, h, d \rangle$ .

**NIDV EDL PVerify** On input  $\langle X_P, X_V, m, \sigma, \pi \rangle$  where  $X_P, X_V \in \mathcal{PK}$ , message  $m \in \mathcal{M}$ , signature  $\sigma \in \mathcal{S}$ , and proof  $\pi = \langle w, r, h, d \rangle \in \mathcal{P}$ , the algorithm computes:

$$\begin{aligned} c &= g^w X_V^r \bmod p \\ G &= g^d X_P^{(h+w)} \bmod p \\ M &= H_1(m)^d \sigma^{(h+w)} \bmod p \end{aligned}$$

and verifies that  $h = H_2(c, G, M, m, \sigma, X_P)$ . If the last equation holds, then the algorithm outputs Accept, otherwise it outputs Reject.

**Comparison to the scheme of Jakobsson et al.** The main difference is that we include the values  $\sigma$  and  $X_P$  in the input of  $H_2$ . Our proof  $\pi$  also has one less element than in the NIDV EDL proof of [13].

#### 7.4 A Concrete NIDV IDL Proof System

The denial proof is an NIDV version of the proof of inequality of discrete logarithms in [4].

Our Setup and KeyGen algorithms are as above in Section 7.3 for the NIDV EDL proof scheme except that now  $\mathcal{P} = G \times \mathbb{Z}_q^4$ .

**NIDV IDL PGen** On input  $\langle x_P, X_V, m, \sigma \rangle$  where  $x_P \in \mathcal{SK}$ ,  $X_V \in \mathcal{PK}$ ,  $m \in \mathcal{M}$ , and  $\sigma \in \mathcal{S}$ , the algorithm picks random  $r \in \mathbb{Z}_q$  and computes  $C = \left(\frac{H_1(m)^{x_P}}{\sigma}\right)^r \bmod p$ .

The algorithm then constructs a designated verifier proof to demonstrate knowledge of some  $\alpha$  and  $\beta$  such that  $C = H_1(m)^\alpha \sigma^{-\beta} \bmod p$  and  $1 = g^\alpha X_P^{-\beta} \bmod p$ . The algorithm sets  $\alpha = x_P r \bmod q$  and  $\beta = r$  for some random  $r \in \mathbb{Z}_q$ , picks random  $r_1, r_2 \in \mathbb{Z}_q$  and computes:

$$\begin{aligned} c &= g^w X_V^r \bmod p \\ G &= g^{r_1} (X_P)^{-r_2} \bmod p \\ M &= H_1(m)^{r_1} (\sigma)^{-r_2} \bmod p \\ h &= H_2(C, c, G, M, m, \sigma, X_P) \\ d_1 &= r_1 - \alpha(h + w) \bmod q \\ d_2 &= r_2 - \beta(h + w) \bmod q \end{aligned}$$

The algorithm outputs  $\pi = \langle C, w, r, h, d_1, d_2 \rangle$  as the NIDV proof to verifier  $V$  that  $\text{DL}(\sigma, H_1(m)) \neq \text{DL}(X_P, g)$ .

**NIDV IDL PVerify** On input  $\langle X_P, X_V, m, \sigma, \pi \rangle$  where  $X_P, X_V \in \mathcal{PK}$ ,  $m \in \mathcal{M}$ ,  $\sigma \in \mathcal{S}$ , and  $\pi = \langle C, w, r, h, d_1, d_2 \rangle \in \mathcal{P}$ , the algorithm first checks that  $C \neq 1$  and then computes:

$$\begin{aligned} c &= g^w X_V^r \bmod p \\ G &= g^{d_1} (X_P)^{-d_2} \bmod p \\ M &= C^{h+w} H_1(m)^{d_1} (\sigma)^{-d_2} \bmod p \end{aligned}$$

and verifies that  $h = H_2(C, c, G, M, m, \sigma, X_P)$ . If the last equation holds, then the algorithm outputs Accept, otherwise it outputs Reject.

## 8 Security of the Concrete Scheme

### 8.1 Security of the NIDV EDL and IDL Proof Systems

**Theorem 1.** *The NIDV EDL proof system of Section 7.3 is correct.*

**Theorem 2.** *The NIDV EDL proof system of Section 7.3 is non-transferable.*

**Theorem 3.** *The NIDV EDL proof system of Section 7.3 is sound in the random oracle model assuming the hardness of the discrete logarithm problem in  $G$ .*

The proof of correctness is trivial and is therefore omitted. The proofs of the other two theorems appear in the Appendix. The proofs of correctness, non-transferability and soundness for the NIDV IDL proof system of Section 7.4 are similar to those for the NIDV EDL proof system, so we omit the details.

### 8.2 Application to the Core Signature Scheme

Since our NIDV EDL and IDL proof systems are secure, they can be composed with our concrete scheme to form secure NIDV confirmation and denial proofs for the NIDV undeniable signature scheme. All that remains for the whole NIDV undeniable signature scheme to be secure is to show that the core signature scheme satisfies the unforgeability and invisibility properties.

**Theorem 4.** *The core signature scheme of Section 7.1 is unforgeable in the random oracle model assuming the hardness of the Computational Diffie-Hellman problem in  $G$ .*

**Theorem 5.** *The core signature scheme of Section 7.1 has invisibility in the random oracle model assuming the hardness of the Decision Diffie-Hellman problem in  $G$ .*

The proof of Theorem 4 is similar to the proof in [17] (corrected in [16]), although we use a slightly different security model. The details are left to the reader. The proof of Theorem 5 is fairly simple and is therefore also left to the reader. Both proofs will appear in the full version of the paper.

Alternative constructions for NIDV EDL and IDL proofs may be possible. For example, it may be the case that the techniques of [8] could yield more general constructions of such NIDV proofs, although we believe that such general constructions are unlikely to be more efficient than the concrete examples presented here.

## 9 Conclusions and Open Problems

We have presented models of security for NIDV proof systems and NIDV undeniable signatures and argued that NIDV proofs can have applications outside of the context of undeniable signatures such as in deniable proofs of knowledge or possession. We then repaired the original NIDV undeniable signature scheme of [13], producing a concrete scheme that is efficient and proven secure.

In future work, it would be interesting to investigate how to extend our model to include strong designated verifier proofs [19, 22, 13] and DVS schemes [15]. It would also be interesting to provide models of security for NIDV versions of confirmer signatures [1, 6, 9, 3] and other signature schemes closely related to undeniable signatures.

## Acknowledgements

We would like to thank Steven Galbraith for valuable comments on this work.

## References

1. J. Boyar, D. Chaum, I. Damgård, and T. P. Pedersen. Convertible undeniable signatures. In A. Menezes and S.A. Vanstone, editors, *Advances in Cryptology – CRYPTO '90*, volume 537 of *LNCS*, pages 189–205. Springer-Verlag, 1991.
2. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

3. J. Camenisch and M. Michels. Confirmer signature schemes secure against adaptive adversaries. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 243–258. Springer-Verlag, 2000.
4. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer-Verlag, 2003.
5. D. Chaum. Zero-knowledge undeniable signatures. In I.B. Damgård, editor, *Advances in Cryptology – EUROCRYPT ’90*, volume 473 of *LNCS*, pages 458–464. Springer-Verlag, 1990.
6. D. Chaum. Designated confirmer signatures. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT ’94*, volume 950 of *LNCS*, pages 86–91. Springer-Verlag, 1994.
7. D. Chaum and H. van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology – CRYPTO ’89*, volume 435 of *LNCS*, pages 212–216. Springer-Verlag, 1990.
8. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO ’94*, volume 893 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 1995.
9. I. Damgård and T. Pedersen. New convertible undeniable signature schemes. In U.M. Maurer, editor, *Advances in Cryptology – EUROCRYPT ’96*, volume 1070 of *LNCS*, pages 372–386. Springer-Verlag, 1996.
10. Y. Desmedt and M. Yung. Weakness of undeniable signature schemes. In D.W. Davies, editor, *Advances in Cryptology – EUROCRYPT ’91*, volume 547 of *LNCS*, pages 205–220. Springer-Verlag, 1991.
11. S. D. Galbraith and W. Mao. Invisibility and anonymity of undeniable and confirmer signatures. In M. Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, page 8097. Springer-Verlag, 2003.
12. M. Jakobsson. Blackmailing using undeniable signatures. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT ’94*, volume 950 of *LNCS*, pages 425–427. Springer-Verlag, 1994.
13. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U.M. Maurer, editor, *Advances in Cryptology – EUROCRYPT ’96*, volume 1070 of *LNCS*, pages 143–154. Springer-Verlag, 1996.
14. F. Laguillaumie and D. Vergnaud. Designated verifier signatures: Anonymity and efficient construction from *any* bilinear map. In C. Blundo and S. Cimato, editors, *SCN 2004*, volume 3352 of *LNCS*, pages 105–119. Springer-Verlag, 2005.
15. H. Lipmaa, G. Wang, and F. Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In L. Caires et al., editor, *Automata, Languages and Programming, ICALP 2005*, volume 3580 of *LNCS*, pages 459–471. Springer-Verlag, 2005.
16. W. Ogata, K. Kurosawa, and S. Heng. The security of the FDH variant of Chaum’s undeniable signature scheme. Cryptology ePrint Archive, Report 2004/290, 2004. Available from <http://eprint.iacr.org/2004/290>.
17. W. Ogata, K. Kurosawa, and S. Heng. The security of the FDH variant of Chaum’s undeniable signature scheme. In S. Vaudenay, editor, *Public Key Cryptography – PKC 2005*, volume 3386 of *LNCS*, pages 328–345. Springer-Verlag, 2005.
18. D. Pointcheval and J. Stern. Security proofs for signature schemes. In U.M. Maurer, editor, *Advances in Cryptology – EUROCRYPT ’96*, volume 1070 of *LNCS*, pages 387–398. Springer-Verlag, 1996.

19. S. Saeednia, S. Kremer, and O. Markowitch. An efficient strong designated verifier signature scheme. In J.I. Lim and D.H. Lee, editors, *Information Security and Cryptology - ICISC 2003*, volume 2971 of *LNCS*, pages 40–54. Springer-Verlag, 2003.
20. R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. In C.S. Laih, editor, *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542. Springer-Verlag, 2003.
21. R. Steinfeld, H. Wang, and J. Pieprzyk. Efficient extension of standard Schnorr/RSA signatures into universal designated-verifier signatures. In F. Bao et al., editor, *PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 86–100. Springer-Verlag, 2004.
22. W. Susilo, F. Zhang, and Y. Mu. Identity-based strong designated verifier signature schemes. In H. Wang et al., editor, *ACISP 2004*, volume 3108 of *LNCS*, pages 313–324. Springer-Verlag, 2004.
23. G. Wang. An attack on not-interactive designated verifier proofs for undeniable signatures. Cryptology ePrint Archive, Report 2003/243, 2003. Available from <http://eprint.iacr.org/>.

## Appendix

*Proof of Theorem 2.* We define algorithm  $A$  as follows. On input  $\langle X_P, x_V, m, \sigma \rangle$ , where  $X_P \in \mathcal{PK}$ ,  $x_V \in \mathcal{SK}$ ,  $(m, \sigma) \in \mathcal{E}$ ,

$A$  chooses random  $d, \alpha, \beta \in \mathbb{Z}_q$  and calculates:

$$\begin{aligned}
 c &= g^\alpha \bmod p \\
 G &= g^d X_P^{-\beta} \bmod p \\
 M &= H_1(m)^d \sigma^{-\beta} \bmod p \\
 h &= H_2(c, G, M, m, \sigma, X_P) \\
 w &= \beta - h \bmod q \\
 r &= (\alpha - w) x_V^{-1} \bmod q
 \end{aligned}$$

$A$  outputs  $\pi = \langle w, r, h, d \rangle$ . It is easy to check that  $\langle X_P, X_V, n, \sigma, \pi \rangle$  will be accepted by PVerify and that  $\pi$  is indistinguishable from any  $\pi' = \langle w', r', h', d' \rangle$  produced by running PGen on input  $\langle x_P, X_V, m', \sigma' \rangle$ .  $\square$

*Proof of Theorem 3.* Suppose that  $H_1$  and  $H_2$  are random oracles and there exists an algorithm  $E$  that makes at most  $\mu_i$  queries to the random oracles  $H_i, i = \{1, 2\}$ , at most  $\mu_s$  Sign queries, and wins the soundness game of Section 4.3 in time at most  $\tau$  with probability at least  $\eta = 10(\mu_s + 1)(\mu_s + \mu_2)/q$ , where  $q$  is exponential in security parameter  $l$ .

We show how to construct an algorithm  $B$  that uses  $E$  to solve the discrete logarithm problem in  $G$ .  $B$  will simulate the random oracles and

the challenger  $C$  in a game with  $E$ .  $B$ 's goal is to solve the discrete logarithm problem on input  $\langle g, X, p, q \rangle$ , that is to find  $x \in \mathbb{Z}_q$  such that  $g^x = X \bmod p$ , where  $g$  is of prime order  $q$  modulo prime  $p$  and generates group  $G$ .

*Simulation:*

$B$  uses the parameters  $\langle g, p, q \rangle$  to run Setup, and gives all the public parameters to  $E$ .  $B$  generates a set of participants  $U$ , where  $|U| = \rho(l)$  and  $\rho$  is a polynomial function of the security parameter  $l$ . For some participant  $J$ ,  $B$  sets  $X_J = X$ , and for each  $I \neq J$ ,  $x_I$  is chosen randomly from  $\mathbb{Z}_q$ , and  $B$  sets  $X_I = g^{x_I} \bmod p$ .  $E$  is given all the public keys  $X_i$ . We define the set of all participants' public keys to be  $\mathcal{X}$ .  $B$  now simulates the challenger by simulating all the oracles which  $E$  can query as follows:

**$H_1$ -Queries:**  $B$  simulates the random oracle by keeping a list of tuples  $\langle M_i, r_i \rangle$  which is called  $L_{H_1}$ . When the oracle is queried with an input  $M \in \{0, 1\}^*$ ,  $B$  responds as follows:

1. If the query  $M$  is already in  $L_{H_1}$  in the tuple  $\langle M, r_i \rangle$ , then  $B$  outputs  $g^{r_i} \bmod p$ .
2. Otherwise  $B$  selects a random  $r \in \mathbb{Z}_q$ , outputs  $g^r \bmod p$  and adds  $\langle M, r \rangle$  to  $L_{H_1}$ .

**$H_2$ -Queries:**  $B$  simulates the  $H_2$  oracle in the same way as  $H_1$  by keeping a list of tuples  $L_{H_2}$ , but the tuples are of the form  $\langle M, s \rangle$ , where  $s$  is chosen randomly from  $\mathbb{Z}_q$ , and the output to a query on  $M$  is  $s$ .

**EGen Queries:**  $E$  can make EGen queries to any oracle  $I$  with public key  $X_I$  on input  $\langle m \rangle$  where  $m \in \mathcal{M}$ . If  $X_I \neq X_J$  then  $B$  runs  $\text{Sign}(x_I, m)$  to produce a signature  $\sigma \in \mathcal{S}$  such that  $(m, \sigma) \in L(X_I)$ . If  $X_I = X_J$  then  $B$  queries  $m$  on the  $H_1$  oracle and receives some  $g^{r_i}$  as response.  $B$  then sets  $\sigma = X_I^{r_i} \bmod p$ .  $B$  outputs  $\langle m, \sigma \rangle$ .

**PGen Queries:**  $E$  can make PGen queries to any oracle  $P$  with public key  $X_P$  on input  $\langle X_V, m, \sigma \rangle$ . If  $X_P \neq X_J$  then  $B$  runs PGen on  $\langle x_P, X_V, m, \sigma \rangle$  and outputs the response. If  $X_P = X_J$  then  $B$  queries  $m$  on  $H_1$  and receives some  $g^{r_i}$ . If  $X_P^{r_i} \bmod p \neq \sigma$  then  $B$  outputs "invalid". Otherwise,  $B$  picks random  $w, r, t, h \in \mathbb{Z}_q$  and computes:

$$\begin{aligned} c &= g^w X_V^r \bmod p \\ G &= g^d X_P^{(h+w)} \bmod p \\ M &= H_1(m)^d \sigma^{(h+w)} \bmod p \end{aligned}$$

If the  $H_2$  oracle has previously been queried on input  $c, G, M, m, \sigma, X_P$ , then  $B$  starts again by picking new  $w, r, t, h$ . Otherwise  $B$  sets  $M =$

“ $c, G, M, m, \sigma, X_P$ ”, adds the tuple  $\langle M, h \rangle$  to  $L_{H_2}$  and outputs  $\pi = \langle w, r, h, d \rangle$ .

**FakePGen Queries:**  $E$  can make FakePGen queries to any oracle  $V$  with public key  $X_V$  on input  $\langle X_P, m, \sigma \rangle$ . If  $X_V \neq X_J$  then  $B$  runs Algorithm  $A$  defined in the proof of Theorem 2 on  $\langle X_P, x_V, m, \sigma \rangle$  and outputs the response. Otherwise  $B$  picks random  $w, r, t, h \in \mathbb{Z}_q$  and computes:

$$\begin{aligned} c &= g^w X_V^r \bmod p \\ G &= g^d X_P^{(h+w)} \bmod p \\ M &= H_1(m)^d \sigma^{(h+w)} \bmod p \end{aligned}$$

If the  $H_2$  oracle has previously been queried on input  $c, G, M, m, \sigma, X_P$ , then  $B$  starts again by picking new  $w, r, t, h$ . Otherwise  $B$  sets  $M = “c, G, M, m, \sigma, X_P”$ , adds the tuple  $\langle M, h \rangle$  to  $L_{H_2}$  and outputs  $\pi = \langle w, r, h, d \rangle$ .

**Corrupt Queries:**  $E$  can make a Corrupt query to any oracle  $I$  with public key  $X_I$ . If  $X_I = X_J$ , then  $B$  aborts and terminates  $E$ . Otherwise  $B$  returns the appropriate private key  $x_I$ .

**Output:** On termination,  $E$  outputs  $\langle X_P^*, X_V^*, m^*, \sigma^*, \pi^* \rangle$ , where  $X_P^*, X_V^* \in \mathcal{X}$ ,  $X_V^*$  is uncorrupted,  $m^* \in \mathcal{M}$ ,  $\sigma^* \in \mathcal{S}$  and  $\pi^* \in \mathcal{P}$ .  $E$  wins if  $\langle X_P^*, X_V^*, m^*, \sigma^*, \pi^* \rangle$  is accepted by PVerify,  $\pi^* = \langle w^*, r^*, h^*, d^* \rangle$  was not the output of some PGen query to  $P^*$  on  $\langle X_V^*, m^*, \sigma^* \rangle$  or some FakePGen query to  $V^*$  on  $\langle X_P^*, m^*, \sigma^* \rangle$ , and either:

1.  $X_P^*$  is uncorrupted, or
2.  $(m^*, \sigma^*) \notin L(X_P^*)$ .

*Case 1.* Suppose that  $X_P^*$  is uncorrupted. If  $h^*$  was not output by any previous PGen or FakePGen query, then by the forking lemma of [18], with a certain probability  $B$  can repeat its simulation so that  $E$  outputs another tuple  $\langle X_P^*, X_V^*, m^*, \sigma^*, \pi \rangle$  where proof  $\pi = \langle w, r, h, d \rangle$  and  $h \neq h^*$ . We then get the equations

$$g^{w^*} X_V^{*r^*} = g^w X_V^r \bmod p \quad (1)$$

$$g^{d^*} X_P^{*(h^*+w^*)} = g^d X_P^{(h+w)} \bmod p \quad (2)$$

$$H_1(m^*)^{d^*} \sigma^{*(h^*+w^*)} = H_1(m^*)^d \sigma^{(h+w)} \bmod p. \quad (3)$$

Now if  $X_V^* \neq X_V$  and  $X_V \neq X_J$ , or  $r^* \neq r$  then  $B$  can solve (1) for the discrete logarithm of  $X_V^*$ . The probability that  $X_V^* = X_J$  is  $\frac{1}{\rho}$ . If  $X_V^* \neq X_V$  and  $X_V = X_J$  then again  $B$  can solve (1) for the discrete logarithm of  $X_V = X_J$ .

If  $X_V^* = X_V$  and  $r^* = r$ , then  $w^* = w$ , and since  $h^* \neq h$  we have that  $h^* + w^* \neq h + w$  so  $B$  can solve (2) for the discrete logarithm of  $X_P^*$ . The probability that  $X_P^* = X_J$  is  $\frac{1}{\rho}$ .

If  $h^*$  was output by some previous PGen query to  $P^*$  on  $\langle X_V, m^*, \sigma^* \rangle$  which produced proof  $\pi = \langle w, r, h^*, d \rangle$ , then since  $h$  and  $h^*$  were outputs from  $H_2$ , with overwhelming probability the inputs to  $H_2$  were identical and we obtain the equations

$$g^{w^*} X_V^{*r^*} = g^w X_V^r \pmod p \quad (4)$$

$$g^{d^*} X_P^{*(h^*+w^*)} = g^d X_P^{*(h^*+w)} \pmod p \quad (5)$$

$$H_1(m^*)^{d^*} \sigma^{*(h^*+w^*)} = H_1(m^*)^d \sigma^{*(h^*+w)} \pmod p. \quad (6)$$

As for equation (1), if  $X_V^* \neq X_V$  or  $r^* \neq r$ , then  $B$  can solve (4) for the discrete logarithm of  $X_J$  with probability  $\frac{1}{\rho}$ .

If  $X_V^* = X_V$  and  $r^* = r$ , then  $w^* = w$ , so  $h^* + w^* = h^* + w$  and therefore  $d^* = d$ . But this means that  $\pi^* = \langle w^*, r^*, h^*, d^* \rangle$  was the output of some PGen query to  $P^*$  on  $\langle X_V^*, m^*, \sigma^* \rangle$ , contradicting our assumption.

If  $h^*$  was output by some previous FakePGen query to  $V^*$  on  $\langle X_P, m^*, \sigma^* \rangle$  which produced proof  $\pi = \langle w, r, h^*, d \rangle$ , then again with overwhelming probability we obtain the equations

$$g^{w^*} X_V^{*r^*} = g^w X_V^{*r} \pmod p \quad (7)$$

$$g^{d^*} X_P^{*(h^*+w^*)} = g^d X_P^{*(h^*+w)} \pmod p \quad (8)$$

$$H_1(m^*)^{d^*} \sigma^{*(h^*+w^*)} = H_1(m^*)^d \sigma^{*(h^*+w)} \pmod p. \quad (9)$$

Now if  $r^* \neq r$  then  $B$  can solve (7) for the discrete logarithm of  $X_V^*$ . The probability that  $X_V^* = X_J$  is  $\frac{1}{\rho}$ . If  $X_P^* \neq X_P$  or  $w^* \neq w$  then  $B$  can solve (8) for the discrete logarithm of  $X_P^*$  (or  $X_P$ ). The probability that  $X_P^* = X_J$  (or  $X_P = X_J$ ) is  $\frac{1}{\rho}$ .

If  $r^* = r$ ,  $X_P^* = X_P$  and  $w^* = w$ , then  $d^* = d$ . But this means that  $\pi^* = \langle w^*, r^*, h^*, d^* \rangle$  was the output of some FakePGen query to  $V^*$  on  $\langle X_P^*, m^*, \sigma^* \rangle$ , contradicting our assumption.

*Case 2.* Suppose now that  $(m^*, \sigma^*) \notin L(X_P^*)$ . If  $h^*$  was not output by any previous PGen or FakePGen query, then by the forking lemma of [18], with a certain probability  $B$  can repeat its simulation so that  $E$  outputs another tuple  $\langle X_P^*, X_V, m^*, \sigma^*, \pi \rangle$  where proof  $\pi = \langle w, r, h, d \rangle$  and  $h \neq h^*$ . We then get the equations

$$g^{w^*} X_V^{*r^*} = g^w X_V^r \pmod p \quad (10)$$

$$g^{d^*} X_P^{*(h^*+w^*)} = g^d X_P^{*(h+w)} \pmod p \quad (11)$$

$$H_1(m^*)^{d^*} \sigma^{*(h^*+w^*)} = H_1(m^*)^d \sigma^{*(h+w)} \pmod p. \quad (12)$$

As for equation (1), if  $X_V^* \neq X_V$  or  $r^* \neq r$  then  $B$  can solve (10) for the discrete logarithm of  $X_J$  with probability  $\frac{1}{\rho}$ .

If  $X_V^* = X_V$  and  $r^* = r$ , then  $w^* = w$ . But since  $h^* \neq h$  we can rewrite equations (11) and (12) as  $X_P^* = g^{\frac{d-d^*}{h^*-h}}$  and  $\sigma^* = H_1(m^*)^{\frac{d-d^*}{h^*-h}}$ . But we assumed that  $(m^*, \sigma^*) \notin L(X_P^*)$ , contradicting our assumption.

Now  $h^*$  cannot have been output by some PGen query to  $P^*$  on  $\langle X_V, m^*, \sigma^* \rangle$  since  $(m^*, \sigma^*) \notin L(X_P^*)$  and the PGen query would have output “invalid” in this case.

If  $h^*$  was output by some previous FakePGen query to  $V^*$  on  $\langle X_P, m^*, \sigma^* \rangle$  which produced proof  $\pi = \langle w, r, h^*, d \rangle$ , then again with overwhelming probability we obtain the equations

$$g^{w^*} X_V^{r^*} = g^w X_V^r \pmod{p} \quad (13)$$

$$g^{d^*} X_P^{(h^*+w^*)} = g^d X_P^{(h^*+w)} \pmod{p} \quad (14)$$

$$H_1(m^*)^{d^*} \sigma^{*(h^*+w^*)} = H_1(m^*)^d \sigma^{*(h^*+w)} \pmod{p}. \quad (15)$$

Now if  $r^* \neq r$  then  $B$  can solve (13) for the discrete logarithm of  $X_V$ .  $X_V = X_J$  with probability  $\frac{1}{\rho}$ . As for (8), if  $X_P^* \neq X_P$  or  $w^* \neq w$  then  $B$  can solve (14) for the discrete logarithm of  $X_J$  with probability  $\frac{1}{\rho}$ .

If  $r^* = r$ ,  $X_P^* = X_P$  and  $w^* = w$ , then  $d^* = d$ . But this means that  $\pi^* = \langle w^*, r^*, h^*, d^* \rangle$  was the output of some FakePGen query to  $V^*$  on  $\langle X_P^*, m^*, \sigma^* \rangle$ , contradicting our assumption.

Since the forking lemma produces a second appropriate signature with expected time at most  $\tau' = 120686\mu_s\tau$ , we find that  $B$  can solve the discrete logarithm problem in time at most  $\tau'$  and with probability at least  $\eta/\rho$ , which is non-negligible, contradicting the hardness of the discrete logarithm problem. □