Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures

Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk

Dept. of Computing, Macquarie University, North Ryde, Australia
{rons,hwang,josef}@comp.mq.edu.au
http://www.ics.mq.edu.au/acac/

Abstract. Universal Designated-Verifier Signature (UDVS) schemes are digital signature schemes with additional functionality which allows *any* holder of a signature to *designate* the signature to any desired *designated-verifier* such that the designated-verifier can verify that the message was signed by the signer, but is unable to convince anyone else of this fact. Since UDVS schemes reduce to standard signatures when no verifier designation is performed, it is natural to ask how to extend the classical Schnorr or RSA signature schemes into UDVS schemes, so that the existing key generation and signing implementation infrastructure for these schemes can be used without modification. We show how this can be efficiently achieved, and provide proofs of security for our schemes in the random oracle model.

1 Introduction

Universal Designated-Verifier Signature (UDVS) schemes introduced by Steinfeld et al [16] are digital signature schemes with additional functionality which allows *any* holder of a signature to *designate* the signature to any desired *designated-verifier* such that the designated-verifier can verify that the message was signed by the signer, but is unable to convince anyone else of this fact, because the verifier's secret key allows him to forge the designated-verifier signatures without the signer's cooperation. Such signature schemes protect the privacy of signature holders from dissemination of signatures by verifiers, and have applications in certification systems [16].

The previous work [16] has shown how to construct efficient deterministic UDVS schemes from Bilinear group-pairs. However, since UDVS schemes reduce to standard signatures when no verifier designation is performed, it is natural to ask how to extend the classical Schnorr [14] or RSA [12] signature schemes into UDVS schemes, so that the existing key generation and signing implementation infrastructure for these schemes can be used without modification — the UDVS functionality can be added to such implementations as an optional feature. In this paper we show how this can be efficiently achieved, and provide concrete proofs of security for our schemes in the random oracle model [2].

As shown in [16], any secure efficient construction of an unconditionally-private UDVS scheme with *unique signatures* (e.g. fully deterministic UDVS schemes with unique secret keys) gives rise to a secure efficient ID-Based Encryption (IBE) scheme. Constructing secure and efficient IBE schemes from classical Diffie-Hellman or RSA problems is a long-standing open problem [3], and until this problem is solved we also cannot hope to construct unconditionally-private UDVS schemes with unique signatures based on classical problems. However, the results in this paper show that by giving up the unique signature requirement and allowing randomization in either the signing (in the case of Schnorr signatures) or designation (in the case of RSA) algorithms, one can construct efficient UDVS schemes from classical problems. Although the UDVS schemes presented in this paper do not have unique signatures, they still achieve perfect unconditional privacy in the sense of [16].

Due to space limitation, the proofs of all theorems in the paper are omitted. They are included in the full version of this paper [17].

1.1 Related Work

As pointed out in [16], the concept of UDVS schemes can be viewed as an application of the general idea of *designated-verifier proofs*, introduced by Jakobsson, Sako and Impagliazzo [8], where a prover non-interactively designates a proof of a statement to a verifier, in such a way that the verifier can simulate the proof by himself with his secret key and thus cannot transfer the proof to convince anyone else about the truth of the statement, yet the verifier himself is convinced by the proof. The distinctive feature of UDVS schemes is *universal* designation: *anyone* who obtains a signature can designate it.

Two of our proposed UDVS schemes (namely $\mathsf{SchUDVS}_2$ and $\mathsf{RSAUDVS}$) make use of the paradigm in [8] of using a trapdoor commitment in a non-interactive proof of knowledge to achieve verifier designation. Since the underlying construction techniques used in these schemes is known, we view our main contribution here is in providing a concrete security analysis which bounds the insecurity of these schemes in terms of the underlying primitives. Our third proposed scheme $\mathsf{SchUDVS}_1$ shows an alternative and more efficient approach than the paradigm of [8], for extending the Schnorr signature scheme into a UDVS scheme, using the Diffie-Hellman function. It is an analogoue of the the bilinear-based approach for constructing UDVS schemes proposed in [16].

Besides providing UDVS schemes based on classical problems, another contribution of this paper is in defining a stronger unforgeability notion for UDVS schemes, which allows the forger access to the attacked designated verifier's verification oracle, as well as to the signer's signing oracle (whereas the model in [16] only allows access to the signing oracle). We analyse our schemes in this stronger model.

Further related work to UDVS schemes is discussed in [16].

2 Preliminaries

2.1 Algorithms and Probability Notation

We say that a function $f : \mathbb{N} \rightarrow \mathbb{R}$ is a *negligible* function if, for any $c > 0$, there exists $k_0 \in \mathbb{N}$ such that $f(k) < 1/k^c$ for all $k > k_0$. We say that a probability function $p : \mathbb{N} \rightarrow \mathbb{R}$ is *overwhelming* if the function $q : \mathbb{N} \rightarrow \mathbb{R}$ defined by $q(k) = 1 - p(k)$ is a negligible function. For various algorithms discussed, we will define a sequence of integers to measure the *resources* of these algorithms (e.g. running-time plus program length, number of oracle queries to various oracles). All these resource parameters can in general be functions of a *security parameter* k of the scheme. We say that an algorithm A with resource parameters $RP = (r_1, \ldots, r_n)$ is *efficient* if each resource parameter $r_i(k)$ of A is bounded by a polynomial function of the security parameter k, i.e. there exists a $k_0 > 0$ and $c > 0$ such that $r_i(k) < k^c$ for all $k > k_0$.

2.2 Discrete-Log and Diffie-Hellman Problems

Our schemes use the following known hard problems for their security. For all these problems GC denotes an algorithm that on input a security parameter k, returns an instance (D_G, g) of a multiplicative group G of prime order q with generator g (the description string D_G determines the group and contains the group order q).

1 Discrete-Log Problem (DL) [4]: Given $(D_G, g) = \mathsf{GC}(k)$ and $y_1 = g^{x_1}$ for uniformly random $x_1 \in \mathbb{Z}_q^*$, compute x_1. We say that DL is *hard* if the success probability $\mathbf{Succ}_{\mathsf{A},\mathsf{DL}}(k)$ of any efficient DL algorithm A with run-time $t(k)$ is upper-bounded by a negligible function $\mathbf{InSec}_{\mathsf{DL}}(t)$ of k.

2 Computational Diffie-Hellman Problem (CDH) [4]: Given $(D_G, g) = \mathsf{GC}(k)$, $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$ for uniformly random $x_1, x_2 \in \mathbb{Z}_q^*$, compute $\mathsf{CDH}_g(g^{x_1}, g^{x_2}) \stackrel{\text{def}}{=} g^{x_1 x_2}$. We say that CDH is *hard* if the success probability $\mathbf{Succ}_{\mathsf{A},\mathsf{CDH}}(k)$ of any efficient CDH algorithm A with run-time $t(k)$ is upper-bounded by a negligible function $\mathbf{InSec}_{\mathsf{CDH}}(t)$ in k.

3 Strong Diffie-Hellman Problem (SDH) [1, 10]: Given $(D_G, g) = \mathsf{GC}(k)$, $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$ for uniformly random $x_1, x_2 \in \mathbb{Z}_q^*$, compute $g^{x_1 x_2}$ given access to a restricted Decision Diffie-Hellman (DDH) oracle $\mathsf{DDH}_{x_1}(.,.)$, which on input $(w, K) \in G \times G$, returns 1 if $K = w^{x_1}$ and 0 else. We say that SDH is *hard* if the success probability $\mathbf{Succ}_{\mathsf{A},\mathsf{SDH}}(k)$ of any efficient SDH algorithm A with run-time $t(k)$ and which makes up to $q(k)$ queries to $\mathsf{DDH}_{x_1}(.,.)$, is upper-bounded by a negligible function $\mathbf{InSec}_{\mathsf{SDH}}(t, q)$ in k.

We remark that the Strong Diffie-Hellman problem (SDH) as defined above and in [1] is a potentially harder variant of the Gap Diffie-Hellman (GDH) problem as defined in [10]. The difference between the two problems is in the DDH oracle: In the GDH problem the DDH oracle accepts *four* inputs (h, z_1, z_2, K) from the attacker and decides whether $K = \mathsf{CDH}_h(z_1, z_2)$, whereas in the SDH problem the attacker can only control the (z_2, K) inputs to the DDH oracle and the other two are fixed to the values $h = g$ and $z_1 = y_1$ (we call this weaker oracle a *restricted* DDH oracle).

2.3 Trapdoor Hash Functions

Some of our proposed UDVS schemes make use of a general cryptographic scheme called a *trapdoor hash function*. We recall the definition and security notions for such schemes [15]. A trapdoor hash function scheme consists of three efficient algorithms: a *key generation* algorithm GKF, a *hash function evaluation* algorithm F, and a *collision solver* algorithm CSF. On input a security parameter k, the (randomized) key-gen. algorithm $\mathsf{GKF}(k)$ outputs a secret/public-key pair (sk, pk). On input a public-key pk, message $m \in M$ and random $r \in R$ (Here M and R are the message and randomness spaces, respectively), the hash function evaluation algorithm outputs a hash string $h = F_{pk}(m; r) \in H$ (here H is the hash string space). On input a key-pair (sk, pk), a message/randomizer pair $(m_1, r_1) \in M \times R$ and a second message $m_2 \in M$, the collision solver algorithm outputs a second randomizer $r_2 = \mathsf{CSF}((sk, pk), (m_1, r_1), m_2) \in R$ such that (m_1, r_1) and (m_2, r_2) constitute a collision for F_{pk}, i.e. $F_{pk}(m_1; r_1) = F_{pk}(m_2; r_2)$.

There are two desirable security properties for a trapdoor hash function scheme $\mathsf{TH} = (\mathsf{GKF}, F, \mathsf{CSF})$. The scheme TH is called *collision-resistant* if the success probability $\mathbf{Succ}^{\mathrm{CR}}_{A,\mathsf{TH}}$ of any efficient attacker A in the following game is negligible. A key-pair $(sk, pk) = \mathsf{GKF}(k)$ is generated, and A is given k and the public-key pk. A can run for time t and succeeds if it outputs a collision (m_1, r_1) and (m_2, r_2) for F_{pk} satisfying $F_{pk}(m_1, r_1) = F_{pk}(m_2, r_2)$ and $m_1 \neq m_2$. We denote by $\mathbf{InSec}^{\mathrm{CR}}_{\mathsf{TH}}(t)$ the maximal success probability in above game over all attackers A with run-time plus program length at most t. The scheme TH is called *perfectly-trapdoor* if it has the following property: for each key-pair $(sk, pk) = \mathsf{GKF}(k)$ and message pair $(m_1, m_2) \in M \times M$, if r_1 is chosen uniformly at random from R, then $r_2 \stackrel{\mathrm{def}}{=} \mathsf{CSF}((sk, pk), (m_1, r_1), m_2) \in R$ has a uniform probability distribution on R.

3 Universal Designated-Verifier Signature (UDVS) Schemes

We review the definition of UDVS schemes and their security notions [16]. For unforgeability we also introduce a stronger notion of security than used in [16].

A Universal Designated Verifier Signature (UDVS) scheme DVS consists of seven algorithms and a 'Verifier Key-Registration Protocol' $\mathsf{P_{KR}}$. All these algorithms may be randomized.

1. **Common Parameter Generation** GC — on input a security parameter k, outputs a string consisting of common scheme parameters cp (publicly shared by all users).
2. **Signer Key Generation** GKS — on input a common parameter string cp, outputs a secret/public key-pair (sk_1, pk_1) for *signer*.
3. **Verifier Key Generation** GKV — on input a common parameter string cp, outputs a secret/public key-pair (sk_3, pk_3) for *verifier*.

4. **Signing** S — on input signing secret key sk_1, message m, outputs *signer*'s publicly-verifiable (PV) signature σ.
5. **Public Verification** V — on input *signer*'s public key pk_1 and message/PV-signature pair (m, σ), outputs verification decision $d \in \{Acc, Rej\}$.
6. **Designation** CDV — on input a *signer*'s public key pk_1, a *verifier*'s public key pk_3 and a message/PV-signature pair (m, σ), outputs a designated-verifier (DV) signature $\widehat{\sigma}$.
7. **Designated Verification** VDV — on input a *signer*'s public key pk_1, *verifier*'s secret key sk_3, and message/DV-signature pair $(m, \widehat{\sigma})$, outputs verification decision $d \in \{Acc, Rej\}$.
8. **Verifier Key-Registration** $\mathsf{P_{KR}} = (\mathsf{KRA}, \mathsf{VER})$ — a protocol between a 'Key Registration Authority' (KRA) and a 'Verifier' (VER) who wishes to register a verifier's public key. On common input cp, the algorithms KRA and VER interact by sending messages alternately from one to another. At the end of the protocol, KRA outputs a pair $(pk_3, Auth)$, where pk_3 is a verifier's public-key, and $Auth \in \{Acc, Rej\}$ is a key-registration authorization decision. We write $P_{KR}(\mathsf{KRA}, \mathsf{VER}) = (pk_3, Auth)$ to denote this protocol's output.

Verifier Key-Reg. Protocol. The purpose of the 'Verifier Key-Registration' protocol is to force the verifier to 'know' the secret-key corresponding to his public-key, in order to enforce the non-transferability privacy property. In this paper we assume, following [16], the *direct* key reg. protocol, in which the verifier simply reveals his secret/public key to the KRA, who authorizes the public-key only if the provided secret-key matches the public key.

3.1 Unforgeability

In the case of a UDVS scheme there are actually two types of unforgeability properties to consider. The first property, called called 'PV-Unforgeability', is just the usual existential unforgeability notion under chosen-message attack [6] for the standard PV signature scheme $D = (\mathsf{GC}, \mathsf{GKS}, \mathsf{S}, \mathsf{V})$ induced by the UDVS scheme (this prevents attacks to fool the *designator*). The second property, called 'DV-Unforgeability', requires that it is difficult for an attacker to forge a DV-signature $\widehat{\sigma}^*$ by the signer on a 'new' message m^*, such that the pair $(m^*, \widehat{\sigma}^*)$ passes the DV-verification test with respect to a given designated-verifier's public key pk_3 (this prevents attacks to fool the designated verifier, possibly mounted by a dishonest designator). As pointed out in [16], it is sufficient to prove the DV unforgeability of a UDVS scheme, since the 'DV-unforgeability' property implies the 'PV-unforgeability' property.

In this paper we introduce a stronger version of DV-unforgeability than used in [16], which we call ST-DV-UF. This model allows the forger also access to the verification oracle of the designated-verifier (this oracle may help the forger because it uses the designated-verifier's secret key, which in turn can be used to forge DV signatures, as required by the privacy property). Note that the model in [16] does not provide this oracle. We believe it is desirable for UDVS schemes to be secure even under such attacks, and place no restrictions on the attacker in accessing the verifier's oracle — in particular the attacker can control both the message/DV sig. pair as well as the signer's public key in accessing this oracle. We remark (proof omitted) that the *strong* DV-unforgeability of the UDVS scheme in [16] follows (in the random-oracle model) from the hardness of a *gap* version of the Bilinear Diffie-Hellman (BDH) problem, in which the attacker has access to a BDH decision oracle (whereas just hardness of BDH suffices for this scheme to achieve the weaker DV-unforgeability notion in [16]).

Definition 1 (Strong DV-Unforgeability). *Let* $\mathsf{DVS} = (\mathsf{GC}, \mathsf{GKS}, \mathsf{GKV}, \mathsf{S}, \mathsf{V}, \mathsf{CDV}, \mathsf{VDV}, \mathsf{P}_{\mathsf{KR}})$ *be a UDVS scheme. Let* A *denote a forger attacking the unforgeability of* DVS*. The Strong DV-Unforgeability notion ST-UF-DV for this scheme is defined as follows:*

1. ***Attacker Input:*** *Signer and Verifier's public-keys* (pk_1, pk_3) *(where* $(sk_1, pk_1) = \mathsf{GKS}(cp)$*,* $(sk_3, pk_3) = \mathsf{GKV}(cp)$ *and* $cp = \mathsf{GC}(k)$*).*
2. ***Attacker Resources:*** *Run-time plus program-length at most* t*, Oracle access to signer's signing oracle* $\mathsf{S}(sk_1, .)$ *(*q_s *queries), oracle access to designated-verifier's verification oracle* $\mathsf{VDV}(., sk_3, ., .)$ *(*q_v *queries) and, if scheme* DVS *makes use of* n *random oracles* $RO_1, \ldots, RO_n$*, allow* q_{RO_i} *queries to the* i*th oracle* RO_i *for* $i = 1, \ldots, n$*. We write attacker's Resource Parameters (RPs) as* $RP = (t, q_s, q_v, q_{RO_1}, \ldots, q_{RO_n})$*.*
3. ***Attacker Goal:*** *Output a forgery message/DV-signature pair* $(m^*, \widehat{\sigma}^*)$ *such that:*
   (1) The forgery is valid, i.e. $\mathsf{VDV}(pk_1, sk_3, m^*, \widehat{\sigma}^*) = Acc$*.*
   (2) Message m^* *is 'new', i.e. has not been queried by attacker to* S*.*
4. ***Security Notion Definition:*** *Scheme is said to be* unforgeable *in the sense of ST-UF-DV if, for any efficient attacker* A*, the probability* $\mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\mathsf{A},\mathsf{DVS}}(k)$ *that* A *succeeds in achieving above goal is a negligible function of* k*. We quantify the insecurity of* DVS *in the sense of ST-UF-DV against arbitrary attackers with resource parameters* $RP = (t, q_s, q_v, q_{RO_1}, \ldots, q_{RO_n})$ *by the probability*

$$\mathbf{InSec}^{\mathrm{ST-UF-DV}}_{\mathsf{DVS}}(t, q_s, q_v, q_{RO_1}, \ldots, q_{RO_n}) \stackrel{\mathrm{def}}{=} \max_{\mathsf{A} \in AS_{RP}} \mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\mathsf{A},\mathsf{DVS}}(k),$$

where the set AS_{RP} *contains all attackers with resource parameters* RP*.*

3.2 Non-Transferability Privacy

Informally, the purpose of the privacy property for a UDVS scheme is to prevent a designated-verifier from using the DV signature σ_{dv} on a message m to produce evidence which convinces a third-party that the message m was signed by the signer. The privacy is achieved because the designated-verifier can forge DV signatures using his secret-key, so even if the designated-verifier reveals his secret key to the third-party, the third-party cannot distinguish whether a DV signature was produced by the designator or forged by the designated-verifier.

We review the privacy model from [16]. The attacker is modelled as a pair of interacting algorithms $(\mathsf{A}_1, \mathsf{A}_2)$ representing the designated-verifier (DV) and

Third-Party (TP), respectively. Let $\widehat{\mathsf{A}_1}$ denote a forgery strategy. The goal of A_2 is to distinguish whether it is interacting with A_1 who has access to designated signatures (game yes) or with $\widehat{\mathsf{A}_1}$, who doesn't have access to designated signatures (game no). More precisely, the game yes runs in two stages as follows.

Stage 1. $(\mathsf{A}_1, \mathsf{A}_2)$ are run on input pk_1, where $(sk_1, pk_1) = \mathsf{GKS}(cp)$ and $cp = \mathsf{GC}(k)$. In this stage, A_1 has access to: (1) signing oracle $\mathsf{S}(sk_1, .)$, (2) KRA key-reg. oracle to register verifier public keys pk via P_{KR} interactions, (3) A_2 oracle for querying a message to A_2 and receiving a response. At end of stage 1, A_1 outputs a message m^* not queried to S during the game (m^* is given to A_2). Let $\sigma^* = \mathsf{S}(sk_1, m^*)$.

Stage 2. A_1 continues to make S,KRA and A_2 queries as in stage 1, but also has access to a designation oracle $\mathsf{CDV}(pk_1, ., m^*, \sigma^*)$ which it can query with any verifier public-key pk which was answered Acc by a previous KRA key-reg. query. At end of stage 2, A_2 outputs a decision $d \in \{\mathsf{yes}, \mathsf{no}\}$.

The game no is defined in the same way except that (1) A_1 is replaced by $\widehat{\mathsf{A}_1}$, (2) $\widehat{\mathsf{A}_1}$ receives as input pk_1 and the program for A_1, (3) $\widehat{\mathsf{A}_1}$ cannot make any designation queries, (4) $\widehat{\mathsf{A}_1}$ makes same number of sign queries as A_1 (possibly 0).

Let P_{yes} and P_{no} denote the probability that A_2 outputs yes in games yes and no, respectively. We let $C_{\widehat{\mathsf{A}_1}}(\mathsf{A}_1, \mathsf{A}_2) \stackrel{\text{def}}{=} |P_{\mathsf{yes}} - P_{\mathsf{no}}|$ denote A_2's distinguishing advantage.

Definition 2. *A UDVS scheme is said to achieve* complete and perfect unconditional privacy *(PR notion) if there exists an efficient forgery strategy* $\widehat{\mathsf{A}_1}$ *such that* $C_{\widehat{\mathsf{A}_1}}(\mathsf{A}_1, \mathsf{A}_2) = 0$ *for any efficient* A_1 *and computationally unbounded* A_2.

4 Two Extensions of Schnorr Signature Scheme into UDVS Schemes

We will present two UDVS schemes which are both extensions of the Schnorr [14] signature scheme (that is, the signer key-generation, signing and public-verification algorithms in both schemes are identical to those of the Schnorr signature). The first UDVS scheme $\mathsf{SchUDVS}_1$ has an efficient and deterministic designation algorithm and its unforgeability relies on the Strong Diffie-Hellman (SDH) assumption. The second UDVS scheme $\mathsf{SchUDVS}_2$ has a less efficient randomized designation algorithm, but its unforgeability follows from the weaker Discrete-Logarithm (DL) assumption (in the random-oracle model).

4.1 First Scheme: $\mathsf{SchUDVS}_1$

Our first UDVS scheme $\mathsf{SchUDVS}_1$ is defined as follows. Let $\{0,1\}^{\leq \ell}$ denote the message space of all bit strings of length at most ℓ bits. The scheme makes use of a cryptographic hash function $H : \{0,1\}^{\leq \ell} \times \{0,1\}^{l_G} \to \{0,1\}^{l_H}$, modelled as a random-oracle [2] in our security analysis. We assume that elements of the group G output by algorithm GC are represented by bit strings of length $l_G \geq l_q$ bits, where $l_q \stackrel{\text{def}}{=} \lfloor \log_2 q \rfloor + 1$ is the bit length of q.

1. **Common Parameter Generation** GC. (Identical to Schnorr). Choose a group G of prime order $q > 2^{l_H}$ with description string D_G (e.g. if G is a subgroup of $\mathbb{Z}_p^*$, the string D_G would contain (p, q)), and let $g \in G$ denote a generator for G. The common parameters are $cp = (D_G, g)$.
2. **Signer Key Generation** GKS. (Identical to Schnorr). Given the common parameters cp, pick random $x_1 \in \mathbb{Z}_q^*$ and compute $y_1 = g^{x_1}$. The public key is $pk_1 = (cp, y_1)$. The secret key is $sk_1 = (cp, x_1)$.
3. **Verifier Key Generation** GKV. Given the common parameters cp, pick random $x_3 \in \mathbb{Z}_q^*$ and compute $y_3 = g^{x_3}$. The public key is $pk_3 = (cp, y_3)$. The secret key is $sk_3 = (cp, x_3)$.
4. **Signing** S. (Identical to Schnorr). Given the signer's secret key (cp, x_1), and message m, choose a random $k \in \mathbb{Z}_q$ and compute $u = g^k$, $r = H(m, u)$ and $s = k + r \cdot x_1 \pmod{q}$. The PV signature is $\sigma = (r, s)$.
5. **Public Verification** V. (Identical to Schnorr). Given the signer's public key y_1 and a message/PV sig. pair $(m, (r, s))$, accept if and only if $H(m, u) = r$, where $u = g^s \cdot y_1^{-r}$.
6. **Designation** CDV. Given the signer's public key y_1, a verifier's public key y_3 and a message/PV-signature pair $(m, (r, s))$, compute $u = g^s \cdot y_1^{-r}$ and $K = y_3^s$. The DV signature is $\widehat{\sigma} = (u, K)$.
7. **Designated Verification** VDV. Given a signer's public key y_1, a verifier's secret key x_3, and message/DV-sig. pair $(m, (u, K))$, accept if and only if $K = (u \cdot y_1^r)^{x_3}$, where $r = H(m, u)$.

Unforgeability. The PV-Unforgeability of $\mathsf{SchUDVS_1}$ is equivalent to the unforgeability of the Schnorr signature, which in turn is equivalent to the Discrete-Logarithm (DL) assumption in G, assuming the random-oracle model for $H(.)$ [11]. However, for the DV-Unforgeability of $\mathsf{SchUDVS_1}$, it is clear that the stronger 'Computational Diffie-Hellman' (CDH) assumption in G is certainly *necessary* — an attacker can forge a DV signature (u, K) on a message m by choosing a random $u \in G$, computing $r = H(m, u)$ and then $K = \mathsf{CDH}_g(u \cdot y_1^r, y_3)$ (indeed this is the idea behind the proof of the privacy of $\mathsf{SchUDVS_1}$ — see below). Moreover, in the strong DV-unforgeability attack setting, the even stronger 'Strong Diffie-Hellman' (SDH) assumption in G is necessary. This is because the forger's access to the verifier's VDV oracle allows him to simulate the fixed-input DDH oracle $\mathsf{DDH}_{x_3}(w, K)$ which decides whether $K = w^{x_3}$ or not (see Sec. 2.2), namely we have $\mathsf{DDH}_{x_3}(w, K) = \mathsf{VDV}(y_1', x_3, m, (u, K))$ with $y_1' = (w \cdot u^{-1})^{r^{-1} \bmod q}$ and $r = H(m, u)$. Note that this does not rule out the possibility that there may be another attack which even bypasses the need to break SDH. Fortunately, the following theorem shows that this is not the case and SDH is also a *sufficient* condition for Strong DV-Unforgeability of $\mathsf{SchUDVS_1}$, assuming the random-oracle model for $H(.)$. The proof uses the forking technique, as used in the proof in [11] of PV-Unforgeability of the Schnorr signature.

Theorem 1 (Strong DV-Unforg. of $\mathsf{SchUDVS_1}$). *If the Strong Diffie-Hellman problem (SDH) is hard in groups generated by the common-parameter algorithm*

GC*, then the scheme* $\mathsf{SchUDVS}_1$ *achieves Strong DV-unforgeability (ST-UF-DV notion) in the random-oracle model for* $H(.)$*. Concretely, the following insecurity bound holds:*

$$\mathbf{InSec}^{\mathrm{ST-UF-DV}}_{\mathsf{SchUDVS}_1}(t, q_s, q_v, q_H) \leq 2\left[(q_H + q_v)\mathbf{InSec}_{\mathsf{SDH}}(t[S], q[S])\right]^{1/2} + \frac{q_s(q_H + q_s + q_v) + 2(q_H + q_v) + 1}{2^{l_H}},$$

where $t[S] = 2t + 2(q_H + q_s + q_v + 1)(T_S + O(l_H)) + (q_s + 1)O(l_q T_g) + O(l_q^2)$*, where* $T_S = O(\log_2(q_H + q_s + q_v) \cdot (\ell + l_G))$ *and* $q[S] = 2q_v$*. Here we denote by* T_g *the time needed to perform a group operation in* G*.*

Privacy. The privacy of $\mathsf{SchUDVS}_1$ follows from the existence of an algorithm for forging DV signatures (with identical probability distribution as that of real DV signatures) using the verifier's secret key, which is a trapdoor for solving the CDH problem on which the DV-Unforgeability relies.

Theorem 2 (Privacy of $\mathsf{SchUDVS}_1$). *The scheme* $\mathsf{SchUDVS}_1$ *achieves complete and perfect unconditional privacy (PR notion).*

4.2 Second Scheme: $\mathsf{SchUDVS}_2$

Our second UDVS scheme $\mathsf{SchUDVS}_2$ trades off efficiency for a better provable unforgeability security guarantee. Rather than using the Diffie-Hellman trapdoor function to achieve privacy, we instead get the designator to produce a Schnorr proof of knowledge of the PV signature (r, s). This proof of knowledge is made non-interactive in the random-oracle model using the Fiat-Shamir heuristic [5], but using a *trapdoor hash function* [9,15] $F_{y_3}(.;.)$ composed with a random oracle $J(.)$ in producing the 'verifier random challenge' $\widehat{r}$ for this proof of knowledge. The designated-verifier's secret key consists of the trapdoor for the hash function F_{y_3}, which suffices for forging the DV signatures, thus providing the privacy property. We remark that a similar technique was used by Jakobsson Sako and Impagliazzo [8], who used a trapdoor commitment scheme in constructing a designated-verifier undeniable signature scheme. Our scheme can use *any* secure trapdoor hash function.

The resulting scheme is defined as follows. Let $\{0,1\}^{\leq \ell}$ denote the message space of all bit strings of length at most ℓ bits. The scheme makes use of two cryptographic hash functions $H : \{0,1\}^{\leq \ell} \times \{0,1\}^{l_G} \rightarrow \{0,1\}^{l_H}$ and $J : \{0,1\}^{\leq \ell} \times \mathbb{Z}_{2^{l_H}} \times \{0,1\}^{l_G} \times \{0,1\}^{l_F} \rightarrow \{0,1\}^{l_J}$, both modelled as random-oracles [2] in our security analysis. We also use a trapdoor hash function scheme $\mathsf{TH} = (\mathsf{GKF}, F, \mathsf{CSF})$ with $F_{y_3} : \{0,1\}^{l_G} \times R_F \rightarrow \{0,1\}^{l_F}$ (we refer the reader to Section 2 for a definition of trapdoor hash function schemes). We assume that elements of the group G output by algorithm GC are represented by bit strings of length $l_G \geq l_q$ bits, where $l_q \stackrel{\mathrm{def}}{=} \lfloor \log_2 q \rfloor + 1$ is the bit length of q.

1. **Common Parameter Generation** GC. (Identical to Schnorr). Choose a group G of prime order q with description string D_G (e.g. if G is a subgroup of $\mathbb{Z}_p^*$, the string D_G would contain (p, q)), and let $g \in G$ denote a generator for G. The common parameters are $cp = (k, D_G, g)$ (k is the security parameter).

2. **Signer Key Generation** GKS. (Identical to Schnorr). Given the common parameters cp, pick random $x_1 \in \mathbb{Z}_q$ and compute $y_1 = g^{x_1}$. The public key is $pk_1 = (cp, y_1)$. The secret key is $sk_1 = (cp, x_1)$.
3. **Verifier Key Generation** GKV. Given the common parameters $cp = k$, run TH's key-gen. algorithm to compute $(sk, pk) = \mathsf{GKF}(k)$. The public key is $pk_3 = (cp, pk)$. The secret key is $sk_3 = (cp, sk, pk)$.
4. **Signing** S. (Identical to Schnorr). Given the signer's secret key (cp, x_1), and message m, choose a random $k \in \mathbb{Z}_q$ and compute $u = g^k$, $r = H(m, u)$ and $s = k + r \cdot x_1 \pmod q$. The PV signature is $\sigma = (r, s)$.
5. **Public Verification** V. (Identical to Schnorr). Given the signer's public key y_1 and a message/PV sig. pair $(m, (r, s))$, accept if and only if $H(m, u) = r$, where $u = g^s \cdot y_1^{-r}$.
6. **Designation** CDV. Given the signer's public key y_1, a verifier's public key $pk_3 = (cp, pk)$ and a message/PV-signature pair $(m, (r, s))$, compute $u = g^s \cdot y_1^{-r}$, $\widehat{u} = g^{\widehat{k}}$ for a random $\widehat{k} \in \mathbb{Z}_q$, $\widehat{h} = F_{pk}(\widehat{u}; \widehat{r}_F)$ for a random $\widehat{r}_F \in R_F$, $\widehat{r} = J(m, r, u, \widehat{h})$ and $\widehat{s} = \widehat{k} + \widehat{r} \cdot s \bmod q$. The DV signature is $\widehat{\sigma} = (u, \widehat{r}_F, \widehat{r}, \widehat{s})$.
7. **Designated Verification** VDV. Given a signer's public key y_1, a verifier's secret key $sk_3 = (cp, sk, pk)$, and message/DV-sig. pair $(m, (u, \widehat{r}_F, \widehat{r}, \widehat{s}))$, accept if and only if $J(m, r, u, \widehat{h}) = \widehat{r}$, where $r = H(m, u)$, $\widehat{h} = F_{pk}(\widehat{u}; \widehat{r}_F)$ and $\widehat{u} = g^{\widehat{s}} \cdot (u \cdot y_1^r)^{-\widehat{r}}$.

Unforgeability. The idea behind the DV-Unforgeability of $\mathsf{SchUDVS}_2$, is that the DV signature is effectively a proof of knowledge of the s portion of the PV Schnorr signature (r, s) by the signer on m. Namely, using the forking technique we can use a forger for $\mathsf{SchUDVS}_2$ to extract s and hence forge a Schnorr PV signature for some unsigned message m, or alternately to break the collision-resistance of the trapdoor hash scheme TH. We have the following concrete result. Note that we need only assume that $J(.)$ is a random-oracle in proving this result, but we provide a count of $H(.)$ queries to allow the use of our reduction bound in conjunction with known results on the unforgeability of the Schnorr signature which assume the random-oracle model for $H(.)$.

Theorem 3 (Strong DV-Unforg. of $\mathsf{SchUDVS}_2$). *If $\mathsf{SchUDVS}_2$ is PV-unforgeable (UF-PV notion) and TH is collision-resistant (CR notion) then $\mathsf{SchUDVS}_2$ achieves Strong DV-unforgeability (ST-UF-DV notion) in the random-oracle model for $J(.)$. Concretely, the following insecurity bound holds:*

$$\begin{aligned}
&\mathbf{InSec}^{\mathrm{ST-UF-DV}}_{\mathsf{SchUDVS}_2}(t, q_s, q_v, q_J, q_H) \leq \\
&\quad 2[(q_J + q_v)q_s]^{1/2} \left[\mathbf{InSec}^{\mathrm{UF-PV}}_{\mathsf{SchUDVS}_2}(t[S], q_s[S], q_H[S]) + \mathbf{InSec}^{\mathrm{CR}}_{\mathsf{TH}}(t[T])\right]^{1/2} \\
&+ \frac{2(q_J + q_v)q_s + 1}{2^{l_J}},
\end{aligned}$$

where $t[S] = t[T] = 2t + O((q_J + q_v)(\ell + l_F + l_G) + l_q T_g + l_q^2)$, $q_s[S] = 2q_s$ and $q_H[S] = 2q_H$. Here we denote by T_g the time needed to perform a group operation in G.

Privacy. The privacy of $\mathsf{SchUDVS}_2$ follows from the existence of an algorithm for forging DV signatures (with identical probability distribution as that of real DV signatures) using the verifier's secret key, which is a trapdoor for solving collisions in TH. In particular we need here the *perfectly-trapdoor* property of TH. This result holds in the standard model (no random-oracle assumptions).

Theorem 4 (Privacy of $\mathsf{SchUDVS}_2$). *If the scheme* TH *is perfectly-trapdoor then* $\mathsf{SchUDVS}_2$ *achieves complete and perfect unconditional privacy (PR notion).*

5 RSA-Based Scheme: RSAUDVS

The idea for the construction of an RSA-based UDVS scheme is analogous to the second Schnorr-based scheme $\mathsf{SchUDVS}_2$, and is described as follows. The PV RSA signature known to the designator is the eth root $\sigma = h^{1/e} \bmod N$ of the message hash h, where (N, e) is the signer's RSA public key. To produce a DV signature on m, the designator computes a zero-knowledge proof of knowledge of the PV signature σ (made non-interactive using Fiat-Shamir method [5]), which is forgeable by the verifier. The Guilliou-Quisquater ID-based signature [7] is based on such a proof and is applied here for this purpose. To make the proof forgeable by the verifier, we use a trapdoor hash function in the computation of the challenge, as done in the $\mathsf{SchUDVS}_2$ scheme. We note that a restriction of the GQ proof that we use is that the random challenge r must be smaller than the public exponent e. To allow for small public exponents and achieve high security level, we apply α proofs in 'parallel', where α is chosen to achieve a sufficient security level — see security bound in our security analysis (a similar technique is used in the Fiat-Shamir signature scheme [5]).

The resulting scheme is defined as follows. Let $\{0,1\}^{\leq \ell}$ denote the message space of all bit strings of length at most ℓ bits. The scheme makes use of two cryptographic hash functions $H : \{0,1\}^{\leq \ell} \times R_S \to \{0,1\}^{l_H}$ and $J : \{0,1\}^{\leq \ell} \times \mathbb{Z}_{l_N}^{\alpha} \times \{0,1\}^{l_F} \to \mathbb{Z}_{2^{l_J/\alpha}}^{\alpha}$. Note that we only need to assume that $J(.)$ is a random-oracle model in our security analysis, and that we allow randomized RSA signatures with hash generation $h = H(m; s)$ for random s. The corresponding verification is to check if $R(h, m) = Acc$ or not, where $R(.)$ is a binary relation function that outputs Acc if h is a valid hash of message m and outputs Rej else. Thus by a suitable choice of $H(.,.)$ and $R(.,.)$ our scheme can instantiated with any of the standardised variants of RSA signatures such as $\mathsf{RSASSA\text{-}PSS}$ or $\mathsf{RSASSA\text{-}PKCS1\text{-}v15}$, as specified in the PKCS1 standard [13]. We also use a trapdoor hash function scheme $\mathsf{TH} = (\mathsf{GKF}, F, \mathsf{CSF})$ with $F_{y_3} : \{0,1\}^{l_G} \times R_F \to \{0,1\}^{l_F}$ (we refer the reader to Section 2 for a definition of trapdoor hash function schemes). Here l_N denotes the length of RSA modulus N of the signer's public key.

1. **Common Parameter Generation** GC. (Identical to RSA). The comm. pars. are $cp = k$ (k is the security parameter).

2. **Signer Key Generation** GKS. (Identical to RSA). Given the common parameters cp, choose a prime $e > 2^{l_J/\alpha}$. Pick random primes p and q such that $N = pq$ has bit-length l_N and $\gcd(e, \phi(N)) = 1$, where $\phi(N) = (p-1)(q-1)$. Compute $d = e^{-1} \bmod \phi(N)$. The public key is $pk_1 = (cp, N, e)$. The secret key is $sk_1 = (cp, N, e, d)$.
3. **Verifier Key Generation** GKV. Given the comm. pars. $cp = k$, run TH's key-gen. algorithm to compute $(sk, pk) = \mathsf{GKF}(k)$. The public key is $pk_3 = (cp, pk)$. The secret key is $sk_3 = (cp, sk, pk)$.
4. **Signing** S. (Identical to RSA). Given the signer's secret key (cp, N, e, d), and message m, choose a random $s \in R_S$ and compute $h = H(m, s)$ and $\sigma = h^d \bmod N$. The PV signature is σ.
5. **Public Verification** V. (Identical to RSA). Given the signer's public key (cp, N, e) and a message/PV sig. pair (m, σ), accept if and only if $R(m, h) = Acc$, where $h = \sigma^e \bmod N$.
6. **Designation** CDV. Given the signer's public key (cp, N, e), a verifier's public key $pk_3 = (cp, pk)$ and a message/PV-signature pair (m, σ), choose α random elements $k_i \in \mathbb{Z}_N^*$ and compute $\widehat{u} = (\widehat{u}_1, \ldots, \widehat{u}_\alpha)$, where $\widehat{u}_i = k_i^e \bmod N$ for $i = 1, \ldots, \alpha$. Compute $\widehat{h} = F_{pk}(\widehat{u}; \widehat{r}_F)$ for random $\widehat{r}_F \in R_F$. Compute $\widehat{r} = (\widehat{r}_1, \ldots, \widehat{r}_\alpha) = J(m, h, \widehat{h})$, where $h = \sigma^e \bmod N$ and $\widehat{r}_i \in \mathbb{Z}_{2^{l_J/\alpha}}$ for $i = 1, \ldots, \alpha$. Compute $\widehat{s} = (\widehat{s}_1, \ldots, \widehat{s}_\alpha)$, where $\widehat{s}_i = k_i \cdot \sigma^{\widehat{r_i}} \bmod N$ for all $i = 1, \ldots, \alpha$. The DV signature is $\widehat{\sigma} = (h, \widehat{r}_F, \widehat{r}, \widehat{s})$.
7. **Designated Verification** VDV. Given a signer's public key (cp, N, e), a verifier's secret key $sk_3 = (cp, sk, pk)$, and message/DV-sig. pair $(m, (h, \widehat{r}_F, \widehat{r}, \widehat{s}))$, accept if and only if $J(m, h, \widehat{h}) = \widehat{r}$ and $R(m, h) = Acc$, where $\widehat{h} = F_{pk}(\widehat{u}; \widehat{r}_F)$ with $\widehat{u} = (\widehat{u}_1, \ldots, \widehat{u}_\alpha)$ and $\widehat{u}_i = \widehat{s}_i^e \cdot h^{-\widehat{r_i}} \bmod N$ for $i = 1, \ldots, \alpha$.

Unforgeability. Similar to the scheme $\mathsf{SchUDVS}_2$, thanks to the soundness of the GQ proof of knowledge of RSA inverses, we can prove the DV unforgeability of RSAUDVS assuming the PV-unforgeability of RSAUDVS (i.e. the existential unforgeability under chosen-message attack of the underlying standard RSA signature (GKS, S, V)) and the collision-resistance of the trapdoor hash TH. The concrete result is the following.

Theorem 5 (Strong DV-Unforg. of RSAUDVS**).** *If* RSAUDVS *is PV-unforgeable (UF-PV notion) and* TH *is collision-resistant (CR notion) then* RSAUDVS *achieves Strong DV-unforgeability (ST-UF-DV notion) in the random-oracle model for $J(.)$. Concretely, the following insecurity bound holds:*

$$\mathbf{InSec}^{\mathrm{ST-UF-DV}}_{\mathsf{RSAUDVS}}(t, q_s, q_v, q_J, q_H) \leq 2[(q_J + q_v)q_s]^{1/2} \left[\mathbf{InSec}^{\mathrm{UF-PV}}_{\mathsf{RSAUDVS}}(t[S], q_s[S], q_H[S]) + \mathbf{InSec}^{\mathrm{CR}}_{\mathsf{TH}}(t[T])\right]^{1/2} + \frac{2(q_J + q_v)q_s + 1}{2^{l_J}},$$

where $t[S] = t[T] = 2t + O((q_J + q_v)(l_F + l_N) + l_e^2 + l_e T_N)$, $q_s[S] = 2q_s$ and $q_H[S] = 2q_H$. Here we denote by T_N the time needed to perform a multiplication in $\mathbb{Z}_N^*$ and $l_e = \log_2(e)$.

Privacy. The privacy of RSAUDVS is unconditional, assuming the perfectly-trapdoor property of the trapdoor hash scheme TH.

Theorem 6 (Privacy of RSAUDVS). *If the scheme* TH *is perfectly-trapdoor then* RSAUDVS *achieves complete and perfect unconditional privacy (PR notion).*

6 Scheme Comparison

The following tables compare the security and performance features of the proposed schemes (also shown for comparison is an entry for the bilinear-based UDVS scheme DVSBM [16]). It is evident that $\mathsf{SchUDVS}_1$ is more computationally efficient than $\mathsf{SchUDVS}_2$ but its security relies on a stronger assumption and it also produces slightly longer DV signatures. The RSA-based scheme RSAUDVS has a disadvantage of long DV signature length, assuming a low public exponent. However, the computation is about the same as in the Schnorr-based schemes.

Scheme	**Extended Sig.**	**Hard Problem**	**Det. Desig?**	**DV Sig. Length** (typ)
$\mathsf{SchUDVS}_1$	Schnorr	SDH	Yes	2.0 kb
$\mathsf{SchUDVS}_2$	Schnorr	DL	No	1.5 kb
RSAUDVS	RSA	RSA	No	11.6 kb
DVSBM	BLS	BDH	Yes	1.0 kb

Table 1. Comparison of UDVS Schemes. The column 'Det Desig?' indicates if the schemes designation algorithm is deterministic. Refer to [17] for assumptions used to compute typical DV sig. lengths.

Scheme	**Desig. Time**	**Ver. Time**
$\mathsf{SchUDVS}_1$	2 exp.	1 exp.
$\mathsf{SchUDVS}_2$	2 exp. + TH	1 exp. + TH
RSAUDVS	$2(\lceil l_J/\log_2(e)\rceil + 1)$ exp. + TH	$\lceil l_J/\log_2(e)\rceil + 1$ exp. + TH
DVSBM	1 pairing	1 pairing + 1 exp.

Table 2. Comparison of UDVS Schemes Approximate Computation Time. Here we count the cost of computing a product $a^x b^y c^z$ as equivalent to a single exponentiation (exp.) in the underlying group. For RSAUDVS exponent lengths are all $\log_2(e)$. TH denotes the cost of evaluating the trapdoor hash function F_{pk} (typ. 1 exp.).

7 Conclusions

We have shown how to efficiently extend the standard Schnorr and RSA signature schemes into Universal Designated-Verifier Signature schemes, and provided a concrete security analysis of the resulting schemes. One problem of our RSA scheme is that the length of designated signatures is larger than standard RSA signatures by a factor roughly proportional to $k/\log_2(e)$, where k is the security parameter and e is the public exponent. An interesting open problem is to find an RSA based UDVS scheme with designated signatures only a constant factor longer than standard RSA signatures, independent of e.

Acknowledgments. The work of Ron Steinfeld, Huaxiong Wang and Josef Pieprzyk was supported by ARC Discovery Grant DP0345366. Huaxiong Wang's work was also supported in part by ARC Discovery Grant DP0344444.

References

1. M. Abdalla, M. Bellare, and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158, Berlin, 2001. Springer-Verlag. See full paper available at `www-cse.ucsd.edu/users/mihir`.
2. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1-st ACM Conf. on Comp. and Comm. Security*, pages 62–73, New York, November 1993. ACM.
3. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, Berlin, 2001. Springer-Verlag.
4. W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Trans. on Information Theory*, 22:644–654, 1976.
5. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Berlin, 1987. Springer-Verlag.
6. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure against Adaptively Chosen Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
7. L.C. Guillou and J.J. Quisquater. A "Paradoxical" Identity-Based Signature Scheme Resulting from Zero-Knowledge. In *CRYPTO '88*, volume 403 of *LNCS*, pages 216–231, Berlin, 1990. Springer-Verlag.
8. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and Their Applications. In *Eurocrypt '96*, volume 1070 of *LNCS*, pages 143–154, Berlin, 1996. Springer-Verlag.
9. H. Krawczyk and T. Rabin. Chameleon Signatures. In *NDSS 2000*, 2000. Available at `http://www.isoc.org/isoc/conferences/ndss/2000/proceedings/`.
10. T. Okamoto and D. Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In *PKC2001*, volume 1992 of *LNCS*, pages 104–118, Berlin, 2000. Springer-Verlag.
11. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. of Cryptology*, 13(3):361–396, 2000.
12. R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–128, 1978.
13. RSA Laboratories. *PKCS♯1 v. 2.1: RSA Cryptography Standard*, 2002.
14. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89*, volume 435 of *LNCS*, pages 239–251, Berlin, 1990. Springer-Verlag.
15. A. Shamir and Y. Tauman. Improved Online/Offline Signature Schemes. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 355–367, Berlin, 2001. Springer-Verlag.
16. R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal Designated-Verifier Signatures. In *Asiacrypt 2003*, volume 2894 of *LNCS*, pages 523–542, Berlin, 2003. Springer-Verlag. Full version available at `http://www.comp.mq.edu.au/~rons`.
17. R. Steinfeld, H. Wang, and J. Pieprzyk. Efficient Extension of Standard Schnorr/RSA signatures into Universal Designated-Verifier Signatures. Cryptology ePrint Archive, Report 2003/193, 2003. `http://eprint.iacr.org/`.