Efficient Extension of Standard Schnorr/RSA signatures into Universal Designated-Verifier Signatures[*]

Ron Steinfeld, Huaxiong Wang, Josef Pieprzyk
Dept. of Computing, Macquarie University, Australia
{rons, hwang, josef}@ics.mq.edu.au

December 15, 2003

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 [26] 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 [26].

The previous work [26] 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 [24] or RSA [22] 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 [26], 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

[*]This is the full version of a paper to appear in PKC 2004.

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 [26].

The proofs of all theorems in the paper are included in the Appendix.

1.1 Related Work

As pointed out in [26], 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 [15], 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 [15] 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 [15], 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 [26].

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 [26] only allows access to the signing oracle). We analyse our schemes in this stronger model.

There have been other approaches proposed to address the privacy threat associated with dissemination of verifiable signed documents. Chaum and van Antwerpen [9, 7] introduced undeniable signatures for this purpose, which require a signer or confirmer's [8, 18, 17, 6, 12] interactive cooperation to verify a signature, but this approach places significant inconvenience and workload on verifiers and confirmers, compared to an off-line non-interactive verification. The work of Brands on *digital credentials* [5, 4] suggests further approaches to enhance user privacy, such as *selective disclosure* of attributes (see also [27]) and *unlinkability* of user transactions. Chameleon signatures [16] allow designation of signatures to verifiers by the *signer*, and in addition allow a signer to prove a forgery by a designated verifier. Ring signatures [21], when restricted to two users, can also be viewed as designated-verifier signatures, where one user is the actual signer and the other user is the designated-verifier who can also forge the two-user ring signature, thus providing the privacy property, called *signer anonymity* in the context of ring signatures. However, signer designation is still performed by the *signer* in these schemes.

2 Preliminaries

2.1 Algorithms and Probability Notation

We say that a function $f : \mathbb{N} \to \mathbb{R}$ 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 $p : \mathbb{N} \to \mathbb{R}$ is *overwhelming* if the function $q : \mathbb{N} \to \mathbb{R}$ 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) [10]: 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) [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 $\mathsf{CDH}_g(g^{x_1}, g^{x_2}) \overset{\text{def}}{=} g^{x_1 x_2}$. 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, 19]: 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 [19]. 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 [25]. 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 $r_2 \overset{\mathrm{def}}{=} \mathsf{CSF}((sk, pk), (m_1, r_1), m_2) \in R$ has a uniform probability distribution on R.

Remark 1. Several examples of trapdoor hash function schemes are given in [25]. To make the UDVS scheme $\mathsf{SchUDVS_2}$ provably secure based on the discrete-log assumption, one can use the following simple trapdoor hash function scheme $\mathsf{TH_{DL}}$, similar to that given in [25]. Let $h : \{0,1\}^* \to \mathbb{Z}_q$ be any collision-resistant hash function. The key gen. algorithm GKF chooses a group G of prime order q and outputs key-pair $sk = (D_G, g, x)$ and $pk = (D_G, g, y)$ where x is random in $\mathbb{Z}_q^*$ and $y = g^x \in G$ (here D_G is description string for group G, including the prime group order q). On input the public key pk, and message/randomizer pair $(m, r) \in \{0,1\}^* \times \mathbb{Z}_q$, the evaluation algorithm computes the hash $h_s = F_{pk}(m; r) = g^{h(m)} \cdot y^r \in G$. On input (sk, pk) and (m_1, r_1, m_2), the collision solver outputs $r_2 = r_1 + (h(m_1) - h(m_2)) \cdot x^{-1} \bmod q$. The scheme $\mathsf{TH_{DL}}$ is collision-resistant as long as h is collision-resistant and the discrete-log is hard in G. $\mathsf{TH_{DL}}$ is also perfectly-trapdoor.

3 Universal Designated-Verifier Signature (UDVS) Schemes

We review the definition of UDVS schemes and their security notions [26]. For unforgeability we also introduce a stronger notion of security than used in [26].

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 [26], the *direct* key reg. protocol, in which the verifier simply reveals his secret key to the KRA.

Consistent UDVS Schemes. The 'PV-Consistency' property requires that the PV-signatures produced by the *signer* are accepted as valid by the PV-verification algorithm V. The 'DV-Consistency' property requires that the DV-signatures produced by the *designator* using the designation algorithm CDV are accepted as valid by the DV-verification algorithm VDV. A UDVS scheme is *consistent* if it has both of the above consistency properties.

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 [13] 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 [26], 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 [26], 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 [26] 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 [26] 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 [26]).

Definition 3.1 (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 [26]. 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.

Definition 3.2 (PR-Privacy[26]). *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* $(\mathsf{A}_1, \mathsf{A}_2)$ *denote an attack pair against the privacy of* DVS. *Let* $\widehat{\mathsf{A}_1}$ *denote a forgery strategy. The privacy notion PR for this scheme is defined as follows:*

1. ***Attacker Input:*** *Signer public-key* pk_1 *(where* $(sk_1, pk_1) = \mathsf{GKS}(cp)$*, and* $cp = \mathsf{GC}(k)$*). Note that* $\widehat{\mathsf{A}_1}$ *also accepts the program for* A_1 *as input.*

2. ***Resources for*** $(\mathsf{A}_1,\widehat{\mathsf{A}_1})$*: Run-time* $(t_1, \widehat{t_1})$*, access to signing oracle* $\mathsf{S}(sk_1, .)$ *(up to* $(q_s, \widehat{q_s})$ *queried messages different from* m^**), access to key-reg. protocol interactions with the* KRA *(up to* $(q_k, \widehat{q_k})$ *interactions), access to* A_2 *oracle (up to* $(q_c, \widehat{q_c})$ *messages). In stage 2,* A_1 *also has access to designation oracle* $\mathsf{CDV}(pk_1, ., m^*, \sigma^*)$ *(up to* q_d *queried keys successfully registered with* KRA*), where* $\sigma^* = \mathsf{S}(sk_1, m^*)$ *is a signer's signature on the challenge message* m^* *output by* A_1 *at end of Stage 1. Note that* $\widehat{\mathsf{A}_1}$ *cannot make any designation queries.*

3. ***Resources for*** A_2*: Run-time* t_2*.*

4. ***Attacker Goal:*** *Let* $P(\mathsf{A}_1, \mathsf{A}_2)$ *and* $P(\widehat{\mathsf{A}_1}, \mathsf{A}_2)$ *denote the probabilities that* A_2 *outputs* yes *when interacting with* A_1 *(game* yes*) and* $\widehat{\mathsf{A}_1}$ *(game* no*), respectively. The goal of* $(\mathsf{A}_1, \mathsf{A}_2)$ *is to achieve a non-negligible convincing measure* $C_{\widehat{\mathsf{A}_1}}(\mathsf{A}_1, \mathsf{A}_2) \overset{\text{def}}{=} |P(\mathsf{A}_1, \mathsf{A}_2) - P(\widehat{\mathsf{A}_1}, \mathsf{A}_2)|$.

5. ***Security Notion Definition:*** *Scheme is said to achieve* privacy *in the sense of PR if there exists an efficient forgery strategy* $\widehat{\mathsf{A}_1}$ *such that the convincing measure* $C_{\widehat{\mathsf{A}_1}}(\mathsf{A}_1, \mathsf{A}_2)$ *achieved by any efficient attacker pair* $(\mathsf{A}_1, \mathsf{A}_2)$ *is negligible in the security parameter* k*. We quantify the insecurity of* DVS *in the sense of PR against arbitrary attacker pairs* $(\mathsf{A}_1, \mathsf{A}_2)$ *with resources* (RP_1, RP_2) *(attacker set* AS_{RP_1,RP_2}*), with respect to arbitrary forgery strategies* $\widehat{A_1}$ *with resources* $\widehat{RP_1}$ *(attacker set* $AS_{\widehat{RP_1}}$*) by the probability*

$$\mathbf{InSec}^{\mathrm{PR}}_{\mathsf{DVS}}(RP_1, \widehat{RP_1}, RP_2) \overset{\text{def}}{=} \min_{\widehat{\mathsf{A}_1} \in AS_{\widehat{RP_1}}} \max_{(\mathsf{A}_1, \mathsf{A}_2) \in AS_{RP_1,RP_2}} C_{\widehat{\mathsf{A}_1}}(\mathsf{A}_1, \mathsf{A}_2).$$

   If $\mathbf{InSec}^{\mathrm{PR}}_{\mathsf{DVS}}(RP_1, \widehat{RP_1}, RP_2) = 0$ *holds for any computationally unbounded* A_2*, it is said to be* perfect unconditional privacy*. If privacy holds when* $\widehat{q_{s1}} = q_{s1}$ *it is said to be* complete *privacy.*

4 Two Extensions of Schnorr Signature Scheme into UDVS Schemes

We will present two UDVS schemes which are both extensions of the Schnorr [24] 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 $l_q \overset{\text{def}}{=} \lfloor \log_2 q \rfloor + 1$ 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 $H(.)$ [20]. 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 [20] of PV-Unforgeability of the Schnorr signature.

Theorem 4.1 (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:

$$\mathbf{InSec}^{\mathrm{ST-UF-DV}}_{\mathsf{SchUDVS_1}}(t, q_s, q_v, q_H) \le 2\left[(q_H + q_v)\mathbf{InSec}_{\mathsf{SDH}}(t[S], q[S])\right]^{1/2} + \frac{q_s(q_H + q_s + q_v) + 2(q_H + q_v) + 1}{2^{l_H}},$$

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 4.2 (Privacy of $\mathsf{SchUDVS_1}$). *The scheme $\mathsf{SchUDVS_1}$ achieves complete and perfect unconditional privacy (in the sense of the PR notion). Concretely:*

$$\mathbf{InSec}^{\mathrm{PR}}_{\mathsf{SchUDVS_1}}(RP_1, \widehat{RP}_1, \infty) = 0, \tag{1}$$

where $RP_1 = (t_1, q_{s1}, q_{c1}, q_{k1}, q_{d1})$ denotes A_1's resource parameters and $\widehat{RP}_1 = (\widehat{t}_1, \widehat{q}_{s1}, \widehat{q}_{c1}, \widehat{q}_{k1})$ denotes the forgery strategy $\widehat{\mathsf{A}_1}$'s resources, which are given by: $\widehat{t}_1 = t_1 + q_{d1}O(l_q T_g + q_{d1} l_G) + q_{k1}O(l_q T_g)$, $\widehat{q}_{s1} = q_{s1}$ (complete privacy), $\widehat{q}_{c1} = q_{c1}$, $\widehat{q}_{k1} = 0$.

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 [11], but using a *trapdoor hash function* [16, 25] $F_{y_3}(.;.)$ 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 [15], 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 $\{0,1\}^{\le \ell}$ denote the message space of all bit strings of length at most ℓ bits. The scheme makes use of two cryptographic hash functions $H : \{0,1\}^{\le \ell} \times \{0,1\}^{l_G} \to \{0,1\}^{l_H}$ and $J : \{0,1\}^{\le \ell} \times \mathbb{Z}_{2^{l_H}} \times \{0,1\}^{l_G} \times \{0,1\}^{l_F} \to \{0,1\}^{l_J}$, 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 $F_{y_3} : \{0,1\}^{l_G} \times R_F \to \{0,1\}^{l_F}$ (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 $l_G \ge l_q$ bits, where $l_q \overset{\text{def}}{=} \lfloor \log_2 q \rfloor + 1$ is the bit length of q.

1. **Common Parameter Generation** GC. (Identical to Schnorr). Choose a group G of prime order q 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 4.3 (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:*

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

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

$$\mathbf{InSec}^{\mathrm{PR}}_{\mathsf{SchUDVS}_2}(RP_1, \widehat{RP}_1, \infty) = 0, \tag{2}$$

where $RP_1 = (t_1, q_{s1}, q_{c1}, q_{k1}, q_{d1})$ *denotes* A_1*'s resource parameters and* $\widehat{RP}_1 = (\widehat{t}_1, \widehat{q}_{s1}, \widehat{q}_{c1}, \widehat{q}_{k1})$ *denotes the forgery strategy* $\widehat{\mathsf{A}_1}$*'s RPs, which are given by:* $\widehat{t}_1 = t_1 + q_{d1}O(l_q T_g + T_F + T_{CSF} + T_J + T_H + q_{d1}l_{pk}) + q_{k1}T_{GKF}$, $\widehat{q}_{s1} = q_{s1}$ *(complete privacy),* $\widehat{q}_{c1} = q_{c1}$, $\widehat{q}_{k1} = 0$. *Here we let* $T_g, T_F, T_{GKF}, T_{CSF}, T_J, T_H$ *denote the time to compute a group operation, and evaluate* F, GKF,CSF, J *and* H *respectively, and* l_{pk} *denotes an upper-bound on the length of public keys for scheme* TH.

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 [11]), which is forgeable by the verifier. The Guilliou-Quisquater ID-based signature [14] 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 [11]).

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 $J : \{0,1\}^{\leq \ell} \times \mathbb{Z}^{\alpha}_{l_N} \times \{0,1\}^{l_F} \to \mathbb{Z}^{\alpha}_{2^{l_J/\alpha}}$. 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 [23]. 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 $\mathsf{RSAUDVS}$ assuming the PV-unforgeability of $\mathsf{RSAUDVS}$ (i.e. the existential unforgeability under chosen-message attack of the underlying standard RSA signature $(\mathsf{GKS}, \mathsf{S}, \mathsf{V})$) and the collision-resistance of the trapdoor hash TH. The concrete result is the following.

Theorem 5.1 (Strong DV-Unforg. of $\mathsf{RSAUDVS}$). *If* $\mathsf{RSAUDVS}$ *is PV-unforgeable (UF-PV notion) and* TH *is collision-resistant (CR notion) then* $\mathsf{RSAUDVS}$ *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 $\mathsf{RSAUDVS}$ is unconditional and complete, assuming the perfectly-trapdoor property of the trapdoor hash scheme TH.

Theorem 5.2 (Privacy of $\mathsf{RSAUDVS}$). *If the scheme* TH *is perfectly-trapdoor then* $\mathsf{RSAUDVS}$ *achieves complete and perfect unconditional privacy (in the sense of the PR notion). Concretely:*

$$\mathbf{InSec}^{\mathrm{PR}}_{\mathsf{RSAUDVS}}(RP_1, \widehat{RP}_1, \infty) = 0, \qquad (3)$$

where $RP_1 = (t_1, q_{s1}, q_{c1}, q_{k1}, q_{d1})$ denotes A_1's resource parameters and $\widehat{RP}_1 = (\widehat{t}_1, \widehat{q}_{s1}, \widehat{q}_{c1}, \widehat{q}_{k1})$ denotes the forgery strategy $\widehat{\mathsf{A}_1}$'s RPs, which are given by: $\widehat{t}_1 = t_1 + q_{d1} O(l_J T_N + T_F + T_{CSF} + T_J + T_H + q_{d1} l_{pk}) + q_{k1} T_{GKF}$, $\widehat{q}_{s1} = q_{s1}$ (complete privacy), $\widehat{q}_{c1} = q_{c1}$, $\widehat{q}_{k1} = 0$. Here we let $T_N, T_F, T_{GKF}, T_{CSF}, T_J, T_H$ denote the time to compute a multiplication modulo N, and evaluate F, GKF, CSF, J and H respectively, and l_{pk} denotes an upper-bound on the length of public keys for scheme TH.

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 [26]. 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 (typically about 10 times the length of a standard RSA signature) assuming a low public exponent $e = 2^{16} + 1$. However, the computation is about the same as in the Schnorr-based schemes. This is because the $O(l_J/\log_2(e))$ exponentiations for RSAUDVS shown in Table 2 use a low exponent e, so the total computation is only $O(l_J)$ modular multiplications.

Scheme	**Extended Sig.**	**Hard Problem**	**Det. Desig?**	**DV Sig. Length** (typ)
$\mathsf{SchUDVS}_1$	Schnorr	SDH	Yes	$2l_G$ (2.0 kb)
$\mathsf{SchUDVS}_2$	Schnorr	DL	No	$l_G + l_F + 2l_q$ (1.5 kb)
RSAUDVS	RSA	RSA	No	$l_N + l_F + l_J + \lceil l_J/\log_2(e)\rceil\, l_N$ (11.6 kb)
DVSBM	BLS	BDH	Yes	l_{G_T} (1.0 kb)

Table 1: Comparison of UDVS Schemes Features. See Secs. 4 and 5 for symbol definitions. The column 'Det Desig?' indicates if the schemes designation algorithm is deterministic or not. The 'DV Sig. Length (typ)' column entries give DV signature lengths for the following typical scheme parameter values. For $\mathsf{SchUDVS}_1$ and $\mathsf{SchUDVS}_2$, we assume a subgroup G of order q in $\mathbb{Z}_p^*$ for p prime, with $l_G = 1.024$ kb, $l_q = l_J = 0.16$ kb. For RSAUDVS, we assume $l_N = 1.024$, $l_J = 0.16$ kb and $e = 2^{16} + 1$. For DVSBM we assume image group of length $l_{G_T} = 1.024$ kb.

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.

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] S. Brands. A technical overview of digital credentials, 1999. Available at `http://www.xs4all.nl/~brands/`.

[5] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates.* MIT Press, 2000.

[6] J. Camenisch and M. Michels. Confirmer Signature Schemes Secure against Adaptive Adversaries. In *Eurocrypt 2000*, volume 1807 of *LNCS*, pages 243–258, Berlin, 2000. Springer-Verlag.

[7] D. Chaum. Zero-Knowledge Undeniable Signatures. In *Eurocrypt '90*, volume 473 of *LNCS*, pages 458–464, Berlin, 1991. Springer-Verlag.

[8] D. Chaum. Designated Confirmer Signatures. In *Eurocrypt '94*, volume 950 of *LNCS*, pages 86–91, Berlin, 1994. Springer-Verlag.

[9] D. Chaum and H. van Antwerpen. Undeniable Signatures. In *Crypto '89*, volume 435 of *LNCS*, pages 212–216, Berlin, 1990. Springer-Verlag.

[10] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Trans. on Information Theory*, 22:644–654, 1976.

[11] 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.

[12] R. Gennaro and T. Rabin. RSA-Based Undeniable Signatures. *J. of Cryptology*, 13:397–416, 2000.

[13] 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.

[14] 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.

[15] 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.

[16] H. Krawczyk and T. Rabin. Chameleon Signatures. In *NDSS 2000*, 2000. Available at `http://www.isoc.org/isoc/conferences/ndss/2000/proceedings/`.

[17] M. Michels and M. Stadler. Generic Constructions for Secure and Efficient Confirmer Signature Schemes. In *Eurocrypt '98*, volume 1403 of *LNCS*, pages 406–421, Berlin, 1998. Springer-Verlag.

[18] T. Okamoto. Designated Confirmer Signatures and Public-Key Encryption are Equivalent. In *Crypto '94*, volume 839 of *LNCS*, pages 61–74, Berlin, 1994. Springer-Verlag.

[19] 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.

[20] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. of Cryptology*, 13(3):361–396, 2000.

[21] R. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In *Asiacrypt 2001*, volume 2248 of *LNCS*, pages 552–565, Berlin, 2001. Springer-Verlag.

[22] 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.

[23] RSA Laboratories. *PKCS♯1 v. 2.1: RSA Cryptography Standard*, 2002.

[24] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89*, volume 435 of *LNCS*, pages 239–251, Berlin, 1990. Springer-Verlag.

[25] A. Shamir and Y. Tauman. Improved Online/Offline Signature Schemes. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 355–367, Berlin, 2001. Springer-Verlag.

[26] 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`.

[27] R. Steinfeld, L. Bull, and Y. Zheng. Content Extraction Signatures. In *International Conference on Information Security and Cryptology ICISC 2001*, volume 2288 of *LNCS*, pages 285–304, Berlin, 2001. Springer-Verlag.

A Proof of Theorem 4.1

We show how to use any efficient forging attacker A for breaking scheme $\mathsf{SchUDVS_1}$ in the sense of ST-UF-DV with non-negligible probability to construct an efficient attacker $\mathsf{A_S}$ for breaking the Strong Diffie-Hellman problem (SDH) with non-negligible probability, thus contradicting the assumed hardness of SDH. More precisely, we show that:

$$\mathbf{Succ}_{\mathsf{A_S},\mathsf{SDH}}(k) \geq (1/4(q_H + q_v)) \cdot \left[\mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\mathsf{A},\mathsf{SchUDVS_1}}(k) - \frac{q_s(q_H + q_s + q_v) + 2(q_H + q_v) + 1}{2^{l_H}}\right]^2, \quad (4)$$

where $\mathsf{A_S}$ has the run-time $t[S]$ and makes $q[S]$ DDH queries, as defined in the theorem statement. The theorem then follows immediately from (4), by taking maximums over all attackers $\mathsf{A_S}$ with the given running time. It remains to construct the attacker $\mathsf{A_S}$ and show that it satisfies (4).

Overview. We convert A into $\mathsf{A_S}$ in two stages. In the first stage, we modify the S oracle simulator $\mathsf{F^S}$ so that it does not use x_1 (by outputting a random pair (r, s) as the signature for message m and modifying the H oracle to answer consistently r whenever $(m, g^s \cdot y_1^{-r})$ is later queried to H). We also modify the VDV oracle simulator $\mathsf{F^{VDV}}$ such that it does not use x_3 directly but only indirectly via queries to a $\mathsf{DDH}_{x_3}(.,.)$ oracle. At the end of the first stage we obtain an efficient algorithm for the following primitive problem $P[q_o, q_d]$:

1. Problem $P[q_o, q_d]$: Given $(D_G, g) = \mathsf{GC}(k)$, $y_1 = g^{x_1}$ and $y_3 = g^{x_3}$ for uniformly random $x_1, x_3 \in \mathbb{Z}_q$, q_o distinct queries $(u[1], \ldots, u[q_o])$ to oracle $O(.)$ (where $r[i] = O(u[i])$ is a uniformly random integer in $\mathbb{Z}_{2^{l_H}}$), and q_d queries to restricted DDH oracle $\mathsf{DDH}_{x_3}(.,.)$, (where $\mathsf{DDH}_{x_1}(w, K)$,

returns 1 if $K = w^{x_1}$ and 0 else), compute $i^* \in \{1, \dots, q_o\}$ and $K^* \in G$ such that $K^* = (u[i^*] \cdot y_1^{r[i^*]})^{x_3}$.

In the second stage we show how to convert any efficient algorithm for primitive problem $P[q_o, q_d]$ into an efficient algorithm for SDH. This reduction is an application of the forking technique [11, 20], which involves running the attacker for $P[q_o, q_d]$ twice, answering its i^*th $O(.)$ query differently in the two runs to obtain two distinct solutions (i^*, K_1^*) (with $K_1^* = (u[i^*] \cdot y_1^{r_1[i^*]})^{x_3}$) and (i^*, K_2^*) (with $K_2^* = (u[i^*] \cdot y_1^{r_2[i^*]})^{x_3}$) for the $P[q_o, q_d]$ instance, from which the solution $g^{x_1 x_3}$ to the SDH instance can be recovered.

We begin with the details of the 'Stage 1' reduction.

Lemma A.1 (Stage 1). *Any ST-UF-DV attacker* A *on scheme* $\mathsf{SchUDVS_1}$ *with resources* (t, q_s, q_v, q_H) *and success probability* $\mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\mathsf{A,SchUDVS_1}}(k)$ *can be converted into an algorithm* $\mathsf{A_P}$ *for problem* $P[q_o, q_d]$ *with run-time* $t[P] = t + (q_s + q_v + q_H + 1)(T_S + O(l_H)) + O(q_s l_q T_g)$ *(where* $T_S = O(log_2(q_s + q_v + q_H)(\ell + l_G)))$*),* $q_o = q_H + q_v$ *O oracle queries,* $q_d = q_v$ *DDH oracle queries, and success probability* $\mathbf{Succ}_{\mathsf{A_P}, P[q_o,q_d]}(k) \geq \mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\mathsf{A,SchUDVS_1}}(k) - [q_s(q_s + q_v + q_H) + 1]/2^{l_H}$.

Proof. Modified Attacker $\widehat{\mathsf{A}}$. We first define a *modified* attacker $\widehat{\mathsf{A}}$ which is obtained from the original attacker A in order to satisfy two properties (which may not be satisfied by A): (1) Each H-query (m_i, u_i) of $\widehat{\mathsf{A}}$ is 'new' (i.e. unequal any earlier query (m_j, u_j) to $H(.)$, made either directly by $\widehat{\mathsf{A}}$ or indirectly by S), and (2) For each VDV-query $(y_{1,i}, m_i, u_i, K_i)$ of $\widehat{\mathsf{A}}$, the pair (m_i, u_i) is not new (i.e. it is equal to an earlier query (m_j, u_j) to $H(.)$ by either $\widehat{\mathsf{A}}$ or S oracle). We obtain $\widehat{\mathsf{A}}$ as follows: $\widehat{\mathsf{A}}$ runs A and stores in a sorted table T the queries (m_i, u_i) and corresponding responses $r_i = H(m_i, u_i)$ to A's and S oracle's H-queries. When A makes an S-query, $\widehat{\mathsf{A}}$ forwards it to the S-oracle and returns the signature (r_i, s_i) to A, adding query (m_i, u_i) and response r_i to table T, where $u_i = g^{s_i} \cdot y_1^{-r_i}$. When A makes a H query (m_i, u_i), $\widehat{\mathsf{A}}$ first searches table T for a matching earlier query (m_j, u_j) and if found, answers $r_i = r_j$ from the table without querying $H(.)$ (if not found, $\widehat{\mathsf{A}}$ queries (m_i, u_i) to H answers response r_i to A, adding (m_i, u_i) and response r_i to T). This modification implies property (1). Similarly, when A makes a VDV oracle query $(y_{1,i}, m_i, u_i, K_i)$, $\widehat{\mathsf{A}}$ searches T for matching earlier query $(m_j, u_j) = (m_i, u_i)$ and if not found, it queries (m_i, u_i) to $H(.)$ and updates T. Then $\widehat{\mathsf{A}}$ forwards $(y_{1,i}, m_i, u_i, K_i)$ to VDV oracle and returns response to A. This modification implies property (2). It is clear that $\widehat{\mathsf{A}}$ has the same success probability as A ($\mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\widehat{\mathsf{A}},\mathsf{SchUDVS_1}} = \mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\mathsf{A,Sch_UDVS_1}}$) and also the same number of S- and VDV- queries ($q_s[\widehat{\mathsf{A}}] = q_s$ and $q_v[\widehat{\mathsf{A}}] = q_v$). However, the H-query bound of $\widehat{\mathsf{A}}$ is larger due to modification (2) ($q_H[\widehat{\mathsf{A}}] = q_H + q_v$), and the running-time of $\widehat{\mathsf{A}}$ is also larger due to modifications (1) and (2) ($t[\widehat{\mathsf{A}}] = t + O((q_v + q_H)\log_2(q_s + q_v + q_H) \cdot (\ell + l_G)) + O(l_q T_g q_s)$), assuming a binary search by $\widehat{\mathsf{A}}$ through the sorted list of past H-queries.

We now consider several attack games, starting at the original attack $\mathsf{Game_0}$ and modifying $\widehat{\mathsf{A}}$'s view simulators to obtain the game $\mathsf{Game_3}$ where $\widehat{\mathsf{A}}$ and its view simulators constitute the desired algorithm $\mathsf{A_P}$ against $P[q_o, q_d]$.

Game$_0$. This is the original forgery attack game UF-DV, where the view simulator $\mathsf{F} = (\mathsf{F^S}, \mathsf{F^H})$ for $\widehat{\mathsf{A}}$ consists of the actual scheme's S and H oracles respectively. The game $\mathsf{Game_0}$ runs as follows on outcome I (we use bold letters to denote random variables which constitute the *view* of $\widehat{\mathsf{A}}$, i.e. the inputs, oracle queries and responses of $\widehat{\mathsf{A}}$, and output of $\widehat{\mathsf{A}}$).

Setup. We run $\widehat{\mathsf{A}}$ on input $\mathbf{I}_A = (\mathbf{D_G}, \mathbf{g}, \mathbf{y}_1, \mathbf{y}_3)$, where $(\mathbf{D_G}, \mathbf{g}) = \mathsf{GC}(k)$,$\mathbf{y}_1 = \mathbf{g}^{x_1}$ for uniformly random $x_1 \in \mathbb{Z}_q$ and $\mathbf{y}_3 = \mathbf{g}^{x_3}$ for uniformly random $x_3 \in \mathbb{Z}_q$.

Oracle Queries. When $\widehat{\mathsf{A}}$ makes its ith oracle query $\mathbf{Q}_i$, F responds as follows:

(1) **S-Query simulator** F^{S}. If $\mathbf{Q}_i = \mathbf{m}_i$ is an S-Query, F^{S} responds with $\mathbf{R}_i = (\mathbf{r}_i, \mathbf{s}_i)$ computed as follows:
$\mathbf{r}_i = H(\mathbf{m}_i, \mathbf{u}_i)$
$\mathbf{s}_i = k_i + \mathbf{r}_i \cdot x_1$
where $u_i = \mathbf{g}^{k_i}$ for uniformly random $k_i \in \mathbb{Z}_q$

(2) H**-Query simulator** F^{H}. If $\mathbf{Q}_i = (\mathbf{m}_i, \mathbf{u}_i)$ is a H-Query, F^{H} responds with $\mathbf{R}_i = \mathbf{r}_i$ defined as follows:

$\mathbf{r}_i = H(\mathbf{m}_i, \mathbf{u}_i)$

(3) VDV-**Query simulator** F^{V}. If $\mathbf{Q}_i = (\mathbf{y}_{1,i}, \mathbf{m}_i, (\mathbf{u}_i, \mathbf{K}_i))$ is a VDV-Query, F^{V} responds with $\mathbf{R}_i = \mathbf{d}_i \in \{\mathsf{Acc}, \mathsf{Rej}\}$ defined as follows:

$\mathbf{d}_i = Acc$ if and only if $\mathbf{K}_i = (\mathbf{u}_i \cdot \mathbf{y}_1^{r_i})^{x_3}$
where $r_i = H(\mathbf{m}_i, \mathbf{u}_i)$

Output. Eventually $\widehat{\mathsf{A}}$ outputs a forgery message/DV-sig. pair $(\mathbf{m}^*, (\mathbf{u}^*, \mathbf{K}^*))$.

This completes the description of Game_0.

Let S_0 denote the event in Game_0 that $\widehat{\mathsf{A}}$ breaks $\mathsf{SchUDVS}_1$ in the sense of ST-UF-DV. By definition, this means:

$$\begin{aligned}\mathsf{S}_0 \;\Rightarrow\; & \text{(a) } \mathbf{K}^* = (\mathbf{u}^* \cdot \mathbf{y}_1^{r^*})^{x_3} \text{ (where } r^* = H(\mathbf{m}^*, \mathbf{u}^*))\\ & \text{(b) } \mathbf{m}^* \neq \mathbf{m}_i \text{ for all } i \in \mathbf{W}^S, \end{aligned} \tag{5}$$

where $\mathbf{W}^S$ denotes the set of S-query indices.

Let Bad_0 denote the event in Game_0 that $\widehat{\mathsf{A}}$ did not make a H-query $(\mathbf{m}^*, \mathbf{u}^*)$ during the attack.

$$\mathsf{Bad}_0 \;\Rightarrow\; (\mathbf{m}_i, \mathbf{u}_i) \neq (\mathbf{m}^*, \mathbf{u}^*) \text{ for all } i \in \mathbf{W}^H, \tag{6}$$

where $\mathbf{W}^H$ denotes the set of H-query indices.

Thanks to the randomness of $H(.)$, we get:

Claim A.1.

$$\Pr[\mathsf{S}_0 \wedge \mathsf{Bad}_0] \leq 1/2^{l_H}.$$

Proof of Claim. If Bad_0 occurs, we know $(\mathbf{m}^*, \mathbf{u}^*)$ is not queried to H by A and also (by property (2) of $\widehat{\mathsf{A}}$ - see definition of $\widehat{\mathsf{A}}$) not queried to H by F^{V}. If S_0 also occurs, we know from (5)(b) that $(\mathbf{m}^*, \mathbf{u}^*)$ is not queried to H by F^{S} either. Hence if $\mathsf{S}_0 \wedge \mathsf{Bad}_0$ occurs, then $(\mathbf{m}^*, \mathbf{u}^*)$ is not queried to H during the game, and therefore $\widehat{\mathsf{A}}$'s view (and hence its output $(\mathbf{K}^*, \mathbf{u}^*)$) is independent of $r^* = H(\mathbf{m}^*, \mathbf{u}^*)$, which is in turn uniformly distributed in $\mathbb{Z}_{2^{l_H}}$. Since the mapping $r^* \mapsto (\mathbf{u}^* \mathbf{y}_1^{r^*})^{x_3}$ is one-to-one over $\mathbb{Z}_{2^{l_H}}$ (because $2^{l_H} < q$ and x_1 and x_3 are non-zero elements of $\mathbb{Z}_q$), there is only one outcome for r^* (out of 2^{l_H} equiprobable outcomes) such that $(\mathbf{u}^* \mathbf{y}_1^{r^*})^{x_3} = \mathbf{K}^*$ is satisfied. The claim follows. □

Let Bad_0^1 denote the event in Game_0 that a $\mathbf{u}_i = g^{k_i}$ used by S-oracle in answering an S-query matches a previous $\mathbf{u}_j$ appearing in an earlier query to H by either $\widehat{\mathsf{A}}$ or S-oracle:

$$\mathsf{Bad}_0^1 \;\Rightarrow\; \text{There exists } i \in \mathbf{W}^S \text{ such that } \mathbf{u}_i = \mathbf{u}_j \text{ for some } j < i. \tag{7}$$

Thanks to the randomness of the k_i's we get:

Claim A.2.

$$\Pr[\mathsf{Bad}_0^1] \leq q_s(q_s + q_v + q_H)/2^{l_H}.$$

Proof of Claim. For each ith S-query, there are at most $q_s[\widehat{\mathsf{A}}] + q_H[\widehat{\mathsf{A}}] = q_s + q_v + q_H$ previous $\mathbf{u}_j$'s that $\mathbf{u}_i$ can collide with. Since $\mathbf{u}_i = g^{k_i}$ is uniformly distributed in G which has order q, $\mathbf{u}_i$ has probability at most $(q_s + q_v + q_H)/q < (q_s + q_v + q_H)/2^{l_H}$ to collide with a previous $\mathbf{u}_j$. Since there are $q_s[\widehat{\mathsf{A}}] = q_s$ S-queries overall, the claim follows. □

Let S_0^2 denote the event in Game_0 that S_0 occurs but neither Bad_0 nor Bad_0^1 occur. That is:

$$\begin{aligned} \mathsf{S}_0^2 \quad \Rightarrow \quad & \text{(a) There exists } \mathbf{i}^* \in \mathbf{W}^H \text{ such that } \mathbf{K}^* = (\mathbf{u}_{\mathbf{i}^*} \cdot \mathbf{y}_1^{\mathbf{r}_{\mathbf{i}^*}})^{x_3} \\ & \;\text{and } (\mathbf{m}_{\mathbf{i}^*}, \mathbf{u}_{\mathbf{i}^*}) = (\mathbf{m}^*, \mathbf{u}^*) \\ & \text{(b) } \mathbf{m}^* \neq \mathbf{m}_i \text{ for all } i \in \mathbf{W}^S \\ & \text{(c) } \mathbf{u}_i \neq \mathbf{u}_j \text{ for all } j < i \text{ and } i \in \mathbf{W}^S. \end{aligned} \tag{8}$$

The above results immediately give:

Claim A.3.

$$\Pr[\mathsf{S}_0^2] \geq \Pr[\mathsf{S}_0] - \frac{q_s(q_s + q_v + q_H) + 1}{2^{l_H}}.$$

Proof of Claim. We have $\Pr[\mathsf{S}_0^2] = \Pr[\mathsf{S}_0 \wedge \neg\mathsf{Bad}_0 \wedge \neg\mathsf{Bad}_0^1] \geq \Pr[\mathsf{S}_0 \wedge \neg\mathsf{Bad}_0] - \Pr[\mathsf{Bad}_0^1] \geq \Pr[\mathsf{S}_0] - \Pr[\mathsf{Bad}_0] - \Pr[\mathsf{Bad}_0^1]$. The claim now follows from Claims A.1 and A.2. □

Game$_1$. In this game we modify the S-oracle simulator F^S for $\widehat{\mathsf{A}}$. The new simulator returns signatures (r_i, s_i) where r_i is just a uniformly random string ρ_i in $\mathbb{Z}_{2^{l_H}}$, without querying (m_i, u_i) to H. The game Game_1 runs as follows.

Setup. We run $\widehat{\mathsf{A}}$ on input $\mathbf{I}_A = (\mathbf{D_G}, \mathbf{g}, \mathbf{y}_1, \mathbf{y}_3)$, where $(\mathbf{D_G}, \mathbf{g}) = \mathsf{GC}(k)$ and $\mathbf{y}_1 = \mathbf{g}^{x_1}$ for uniformly random $x_1 \in \mathbb{Z}_q$ and $\mathbf{y}_3 = \mathbf{g}^{x_3}$ for uniformly random $x_3 \in \mathbb{Z}_q$.

Oracle Queries. When $\widehat{\mathsf{A}}$ makes its ith oracle query $\mathbf{Q}_i$, F responds as follows:

(1) S**-Query simulator** F^S. If $\mathbf{Q}_i = \mathbf{m}_i$ is an S-Query, F^S responds with $\mathbf{R}_i = (\mathbf{r}_i, \mathbf{s}_i)$ computed as follows:
   $\mathbf{r}_i = \rho_i$ for uniformly random $\rho_i \in \mathbb{Z}_{2^{l_H}}$
   $\mathbf{s}_i = k_i + \mathbf{r}_i \cdot x_1$
   where $u_i = \mathbf{g}^{k_i}$ for uniformly random $k_i \in \mathbb{Z}_q$

(2) H**-Query simulator** F^H. As in Game_0.

(3) VDV**-Query simulator** F^V. As in Game_0.

Output. Eventually $\widehat{\mathsf{A}}$ outputs a forgery message/DV-sig. pair $(\mathbf{m}^*, (\mathbf{u}^*, \mathbf{K}^*))$.

This completes the description of Game_1. Let S_1^2 denote the event in Game_1 corresponding to S_0^2 in Game_0 (i.e. S_1^2 is also defined by (8) over the view of $\widehat{\mathsf{A}}$).

Claim A.4.

$$\Pr[\mathsf{S}_1^2] \geq \Pr[\mathsf{S}_0^2].$$

Proof of Claim. For outcomes in S_0^2 in Game_0, the $\mathbf{u}_i$ for each ith S-query is 'new' and hence by randomness of $H(.)$ the responses $\mathbf{r}_i = H(\mathbf{m}_i, \mathbf{u}_i)$ are uniform and independent in $\mathbb{Z}_{2^{l_H}}$. Furthermore, $H(.)$ is never queried again at $(\mathbf{m}_i, \mathbf{u}_i)$ during the game. Thus for each outcome in S_0^2 in Game_0 there is a corresponding outcome in S_1^2 in Game_1, where the outcomes for ρ_i in Game_1 coincide with outcomes for $H(\mathbf{m}_i, \mathbf{u}_i)$ in Game_0 for all $i \in \mathbf{W}^S$, maintaining the view of A unchanged. Since the ρ_i's are uniform and independent in $\mathbb{Z}_{2^{l_H}}$ this outcome has the same probability in Game_1 as the original outcome in Game_0. The claim follows. □

Game$_2$. In this game we further modify the S-oracle simulator F^{S} for $\widehat{\mathsf{A}}$. The new simulator returns signatures (r_i, s_i) where s_i set equal to a uniformly random element ζ_i in $\mathbb{Z}_q$, thus eliminating the use of x_1 by F^{S}.

Setup. We run $\widehat{\mathsf{A}}$ on input $\mathbf{I}_A = (\mathbf{D_G}, \mathbf{g}, \mathbf{y}_1, \mathbf{y}_3)$, where $(\mathbf{D_G}, \mathbf{g}) = \mathsf{GC}(k)$ and $\mathbf{y}_1 = \mathbf{g}^{x_1}$ for uniformly random $x_1 \in \mathbb{Z}_q$ and $\mathbf{y}_3 = \mathbf{g}^{x_3}$ for uniformly random $x_3 \in \mathbb{Z}_q$.

Oracle Queries. When $\widehat{\mathsf{A}}$ makes its ith oracle query $\mathbf{Q}_i$, F responds as follows:

(1) S-**Query simulator** F^{S}. If $\mathbf{Q}_i = \mathbf{m}_i$ is an S-Query, F^{S} responds with $\mathbf{R}_i = (\mathbf{r}_i, \mathbf{s}_i)$ computed as follows:
$\mathbf{r}_i = \rho_i$ for uniformly random $\rho_i \in \mathbb{Z}_{2^{l_H}}$
$\mathbf{s}_i = \zeta_i$ for uniformly random $\zeta_i \in \mathbb{Z}_q$

(2) H-**Query simulator** F^{H}. As in Game_0.

(3) VDV-**Query simulator** F^{V}. As in Game_0.

Output. Eventually $\widehat{\mathsf{A}}$ outputs a forgery message/DV-sig. pair $(\mathbf{m}^*, (\mathbf{u}^*, \mathbf{K}^*))$.

This completes the description of Game_2. Let S_2^2 denote the event in Game_2 corresponding to S_1^2 in Game_1 (i.e. S_2^2 is also defined by (8) over the view of $\widehat{\mathsf{A}}$).

Claim A.5.

$$\Pr[\mathsf{S}_2^2] \geq \Pr[\mathsf{S}_1^2].$$

Proof of Claim. In Game_1, the $\mathbf{s}_i = k_i + \mathbf{r}_i \cdot x_1 \bmod q$ signature portions are uniformly random in $\mathbb{Z}_q$ and independent of $\mathbf{r}_i$, thanks to the uniformly random and independent choice of $k_i \in \mathbb{Z}_q$ and the fact that mapping $k_i \mapsto k_i + \mathbf{r}_i \cdot x_1 \bmod q$ is a permutation of $\mathbb{Z}_q$. Hence, in Game_2, setting $\mathbf{s}_i = \zeta_i$ for uniformly random and independent $\zeta_i \in \mathbb{Z}_q$ leaves the view of $\widehat{\mathsf{A}}$ unchanged and immediately gives the claimed result. □

Game$_3$. In this game we modify the VDV-oracle simulator F^{V} for $\widehat{\mathsf{A}}$ to use x_3 only indirectly via calls to a $\mathsf{DDH}_{x_3}(.,.)$ oracle. We also eliminate calls to $H(.)$ by F^{V}, using the fact that $\widehat{\mathsf{A}}$ never makes 'new' $(\mathbf{m}_i, \mathbf{u}_i)$ queries to VDV.

Setup. We run $\widehat{\mathsf{A}}$ on input $\mathbf{I}_A = (\mathbf{D_G}, \mathbf{g}, \mathbf{y}_1, \mathbf{y}_3)$, where $(\mathbf{D_G}, \mathbf{g}) = \mathsf{GC}(k)$ and $\mathbf{y}_1 = \mathbf{g}^{x_1}$ for uniformly random $x_1 \in \mathbb{Z}_q$ and $\mathbf{y}_3 = \mathbf{g}^{x_3}$ for uniformly random $x_3 \in \mathbb{Z}_q$.

Oracle Queries. When $\widehat{\mathsf{A}}$ makes its ith oracle query $\mathbf{Q}_i$, F responds as follows:

(1) S-**Query simulator** F^{S}. As in Game_2.

(2) H-**Query simulator** F^{H}. As in Game_0.

(3) VDV-**Query simulator** F^{V}. If $\mathbf{Q}_i = (\mathbf{y}_{1,i}, \mathbf{m}_i, (\mathbf{u}_i, \mathbf{K}_i))$ is a VDV-Query, F^{V} responds with $\mathbf{R}_i = \mathbf{d}_i \in \{\mathsf{Acc}, \mathsf{Rej}\}$ defined as follows:

$\mathbf{d}_i = Acc$ if and only if $\mathsf{DDH}_{x_3}(\mathbf{u}_i \cdot \mathbf{y}_1^{r_i}, \mathbf{K}_i) = 1$
where $r_i = \mathbf{r}_j$ and $j \in \mathbf{W}^S \vee \mathbf{W}^H$ is the index
of a previous matching query $(\mathbf{m}_j, \mathbf{u}_j) = (\mathbf{m}_i, \mathbf{u}_i)$

Output. Eventually $\widehat{\mathsf{A}}$ outputs a forgery message/DV-sig. pair $(\mathbf{m}^*, (\mathbf{u}^*, \mathbf{K}^*))$.

This completes the description of Game_3. Let S_3^2 denote the event in Game_3 corresponding to S_2^2 in Game_2 (i.e. S_3^2 is also defined by (8) over the view of $\widehat{\mathsf{A}}$).

Claim A.6.

$$\Pr[\mathsf{S}_3^2] \geq \Pr[\mathsf{S}_2^2].$$

Proof of Claim. In Game_2, the test by F^{V} if $(\mathbf{u}_i \cdot \mathbf{y}_1^{r_i})^{x_3} = \mathbf{K}_i$ or not, is equivalent to testing if $\mathsf{DDH}_{x_3}(\mathbf{u}_i \cdot \mathbf{y}_1^{r_i}, \mathbf{K}_i) = 1$, as used in Game_3. The method of computing $\mathbf{r}_i$ in Game_3 is equivalent to querying $(\mathbf{m_i}, \mathbf{u}_i)$ to $H(.)$ because by construction of $\widehat{\mathsf{A}}$, $(\mathbf{m_i}, \mathbf{u}_i)$ must have previously appeared in a H- or S- query. $\square$

So in Game_3, $\widehat{\mathsf{A}}$ with its view simulators constitutes an algorithm $\mathsf{A_P}$ for problem $P[q_o, q_d]$, which on input $(D_G, g, y_1 = g^{x_1}, y_3 = g^{x_3})$, makes $q_o \overset{\text{def}}{=} q_H[\widehat{\mathsf{A}}] = q_H + q_v$ distinct queries $\mathbf{u}_i$ to a random oracle $O(.)$ (the $H(.)$-queries of F^{H})receiving answers $\mathbf{r}_i$ uniform in $\mathbb{Z}_{2^{l_H}}$, and $q_d \overset{\text{def}}{=} q_v[\widehat{\mathsf{A}}] = q_v$ queries to the $\mathsf{DDH}_{x_3}(.,.)$ oracle (the DDH queries of F^{V}), and outputs $(\mathbf{i}^*, \mathbf{K}^*)$ such that $\mathbf{K}^* = (\mathbf{u}_{\mathbf{i}^*} \cdot \mathbf{y}_1^{\mathbf{r}_{\mathbf{i}^*}})^{x_3}$ with probability at least $\Pr[\mathsf{S}_3^2] \geq \mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\mathsf{A},\mathsf{SchUDVS}_1}(k) - [q_s(q_s+q_v+q_H)+1]/2^{l_H}$, using claims A.1 to A.6. The run-time of $\mathsf{A_P}$ is the sum of the run-time $t[\widehat{\mathsf{A}}] = t + O((q_v+q_H)\log_2(q_s+q_v+q_H)\cdot(\ell+l_G)) + O(l_q T_g q_s)$ of $\widehat{\mathsf{A}}$, the run-time $t[\mathsf{F}^{\mathsf{V}}] = O((q_v+q_H)\log_2(q_s+q_v+q_H)\cdot(\ell+l_G)) + O(l_q T_g q_s)$ of F^{V} (assuming a binary search is used as for obtaining $\widehat{\mathsf{A}}$ from A), and the time $O(\log_2(q_s+q_v+q_H)\cdot(\ell+l_G))$ for $\mathsf{A_P}$ to search for $\mathbf{i}^*$ such that $(\mathbf{m}^*, \mathbf{u}^*) = (\mathbf{m}_{\mathbf{i}^*}, \mathbf{u}_{\mathbf{i}^*})$. This establishes the lemma. $\square$

We will need the following lemma [20] in the analysis of the forking technique for the 'Stage 2' reduction. A proof can be found in [20] but we provide one here also for completeness.

Lemma A.2 (Splitting Lemma[20]). *Let a and b denote independent random variables over finite sets A and B, respectively, with probability mass functions $P_A(.)$ and $P_B(.)$, respectively. Let $S \subseteq A \times B$ be a set with $\Pr[S] \geq \epsilon$. For each $a \in A$, let $S(a) \subseteq B$ denote the set of $b \in B$ such that $(a,b) \in S$. Then there exists a 'good' subset G of S such that:*

$$\Pr[G] \geq \epsilon/2$$

and, for all $(a,b) \in G$,

$$\Pr[S(a)] \geq \epsilon/2.$$

Proof. Let us *define* the good set G to be the set of all $(a,b) \in S$ such that $\Pr[S(a)] \geq \epsilon/2$. Then it is enough to show that $\Pr[G] \geq \epsilon/2$.

Suppose, towards a contradiction, that $\Pr[G] < \epsilon/2$. Then $\Pr[S] = \Pr[G] + \Pr[S \wedge \neg G] < \epsilon/2 + \Pr[S \wedge \neg G]$. But $(a,b) \in S \wedge \neg G$ means that $a \in W_A$, where $W_A \subseteq A$ is the set of $a \in A$ such that $\Pr[S(a)] < \epsilon/2$. So $\Pr[S \wedge \neg G] = \sum_{a \in W_A} \sum_{b \in S(a)} P_A(a) P_B(b) = \sum_{a \in W_A} P_A(a) \cdot \Pr[S(a)] < \epsilon/2$ since $\sum_{b \in S(a)} P_B(b) = \Pr[S(a)] < \epsilon/2$ for all $a \in W_A$. It follows that $\Pr[S] < \epsilon/2 + \epsilon/2 = \epsilon$, a contradiction. This shows that $\Pr[G] \geq \epsilon/2$, which completes the proof. $\square$

We will also use the following inequality.

Lemma A.3. *Let $p = \sum_{j=1}^q p_j$ for some q real numbers $p_1, \ldots, p_q$ and let $\delta > 0$ be given. If $p \geq q \cdot \delta$ then the following inequality holds:*

$$\sum_{j=1}^{q} p_j \cdot (p_j - \delta) \geq (1/q) \cdot (p - q \cdot \delta)^2.$$

Proof. We have $\sum_{j=1}^q p_j \cdot (p_j - \delta) = \sum_{j=1}^q p_j^2 - p \cdot \delta$. Using the Cauchy-Schwartz inequality we have $\sum_{j=1}^q p_j^2 \geq (1/q) \cdot (\sum_{j=1}^q p_j)^2 = (1/q) \cdot p^2$, so $\sum_{j=1}^q p_j \cdot (p_j - \delta) \geq (1/q)(p^2 - q \cdot \delta \cdot p)$. But from the assumption that $p \geq q\cdot\delta$, we have $(q\cdot\delta)p \geq (q\cdot\delta)^2$ and hence $p^2 - q\cdot\delta\cdot p \geq p^2 - 2(q\cdot\delta)p + (q\cdot\delta)^2 = (p - q\cdot\delta)^2$, which gives the claimed inequality. $\square$

Lemma A.4 (Stage 2). *Any algorithm* $\mathsf{A_P}$ *for problem* $P[q_o, q_d]$ *with run-time* $t[P]$ *and success probability* $\mathbf{Succ}_{\mathsf{A_P},P[q_o,q_d]}(k) \geq 2q_o/2^{l_H}$ *can be converted into an algorithm* $\mathsf{A_S}$ *for* SDH *with run-time* $t[S] = 2t[P] + O(l_q^2 + l_q T_g)$ *which makes* $q[S] = 2q_d$ DDH *queries, and has success probability* $\mathbf{Succ}_{\mathsf{A_S},\mathsf{SDH}}(k) \geq (1/q_o)\cdot[\mathbf{Succ}_{\mathsf{A_P},P[q_o,q_d]}(k)/2 - q_o/2^{l_H}]^2$*. Here* T_g *denotes the time to perform a group operation in* G*.*

Proof. Given SDH input instance $(D_G, g, y_1 = g^{x_1}, y_2 = g^{x_2})$, our SDH algorithm $\mathsf{A_S}$ works as follows (we assume without loss of generality that $\mathsf{A_S}$'s DDH oracle is $\mathsf{DDH}_{x_2}(.,.)$).

Setup. $\mathsf{A_S}$ first sets up two random vectors $\overrightarrow{r} = (r[1], \ldots, r[q_o])$ and $\overrightarrow{\widehat{r}} = (\widehat{r}[1], \ldots, \widehat{r}[q_o])$ with $r[i]$'s and $\widehat{r}[i]$'s chosen uniformly and independently at random from $\mathbb{Z}_{2^{l_H}}$ (these vectors will be used to answer $\mathsf{A_P}$'s $O(.)$ queries).

First Run. $\mathsf{A_S}$ runs $\mathsf{A_P}$ on input $(D_g, g, y_1, y_2; \omega)$, where ω is a random bit string used as the randomness input of $\mathsf{A_P}$ (if $\mathsf{A_P}$ is randomized), and answers its oracle queries as follows:

(1) $O(.)$**-Query simulator** $\mathsf{F^O}$. When $\mathsf{A_P}$ makes its ith $O(.)$ query $u[i]$, $\mathsf{A_S}$ responds with $r[i]$.

(2) $\mathsf{DDH}_{x_2}(.,.)$**-Query simulator** $\mathsf{F^D}$. When $\mathsf{A_P}$ makes its ith $\mathsf{DDH}_{x_2}(.,.)$ query $(w[i], K[i])$, $\mathsf{A_S}$ simply forwards the query to its $\mathsf{DDH}_{x_2}(.,.)$ oracle and forwards the oracle's response $d[i]$ to $\mathsf{A_P}$.

First Run Output. At the end of first run, $\mathsf{A_P}$ outputs (i^*, K^*) (note that if this run is successful then $K^* = (u[i^*]\cdot y_1^{r[i^*]})^{x_2}$).

Second Run. $\mathsf{A_S}$ runs $\mathsf{A_P}$ again on the *same* input $(D_g, g, y_1, y_2; \omega)$ as used in first run, but answers its oracle queries differently as follows:

(1) $O(.)$**-Query simulator** $\mathsf{F^O}$. When $\mathsf{A_P}$ makes its ith $O(.)$ query $u[i]$, $\mathsf{A_S}$ responds with $r[i]$ for $i < i^*$ and with $\widehat{r}[i]$ for $i \geq i^*$.

(2) $\mathsf{DDH}_{x_2}(.,.)$**-Query simulator** $\mathsf{F^D}$. Answered as in first run.

Second Run Output. At the end of second run, $\mathsf{A_P}$ outputs $(\widehat{i^*}, \widehat{K^*})$ (note that if this run is successful and $\widehat{i^*} = i^*$ then $\widehat{K^*} = (u[i^*]\cdot y_1^{\widehat{r}[i^*]})^{x_2}$).

$\mathsf{A_S}$*'s output.* $\mathsf{A_S}$ computes and returns an estimate $K \overset{\text{def}}{=} (\widehat{K^*}/K^*)^{(\widehat{r}[i^*]-r[i^*])^{-1} \bmod q}$ for the SDH instance solution $g^{x_1x_2}$ (if $\widehat{r}[i^*] = r[i^*]$ then $\mathsf{A_S}$ fails).

This completes the description of $\mathsf{A_S}$. The running-time of $\mathsf{A_S}$ is twice the run-time of $\mathsf{A_P}$ plus the time to compute K at the end, which can be done in time $O(l_q^2 + l_q\cdot T_g)$. The number of $\mathsf{DDH}_{x_2}(.,.)$ queries made by $\mathsf{A_S}$ is up to twice the number of queries made by $\mathsf{A_P}$. This establishes the claimed resources of $\mathsf{A_S}$.

We now lower bound the success probability of $\mathsf{A_S}$. For $i \in \{1, \ldots, q_o\}$, we call a run of $\mathsf{A_P}$ i-successful if $\mathsf{A_P}$ succeeds and $i^* = i$. Note that if both first and second runs of $\mathsf{A_P}$ are i-successful for some i, then we have $i^* = \widehat{i^*} = i$, $K^* = (u[i]\cdot y_1^{r[i]})^{x_2}$ and $\widehat{K^*} = (u[i]\cdot y_1^{\widehat{r[i]}})^{x_2}$ (note that $u[1], \ldots, u[i]$ are the same in both runs because the view of $\mathsf{A_P}$ is the same up to ith $O(.)$ query response) and consequently $\mathsf{A_S}$'s estimate K is correct (whenever $r[i] \neq \widehat{r}[i]$ so $(\widehat{r}[i] - r[i])^{-1} \bmod q$ exists) because $K = [(u[i]\cdot y_1^{r[i]})^{x_2}/(u[i]\cdot y_1^{\widehat{r}[i]})^{x_2}]^{(\widehat{r}[i]-r[i])^{-1} \bmod q} = y_1^{x_2} = g^{x_1x_2}$.

So it remains to lower bound the probability of the event S^* that both runs of $\mathsf{A_P}$ are i-successful for some $i \in \{1, \ldots, q_o\}$ and $\widehat{r}[i] \neq r[i]$. To do this, we split S^* into q_o disjoint subevents S^*_i according the value of i and bound each one. For each i, let A_i denote the outcome space for the random variable $a_i = (D_g, g, y_1, y_2, \omega, r[1], \ldots, r[i-1])$ consisting of the view of A^{P} up to the ith query to $O(.)$, and let B_i denote the outcome space for the independent random variable $b_i = (r[i], \ldots, r[q_o])$ consisting of the view of A^{P} after the ith query to $O(.)$ (including the response $r[i]$ to the ith query). Note that the event $\mathsf{S_i}$ that a run of $\mathsf{A_P}$ is i-successful is a subset of $A_i \times B_i$ with probability $p_i \overset{\text{def}}{=} \Pr[(a_i, b_i) \in \mathsf{S_i}]$. Applying the Splitting Lemma A.2, we know that there exists a subevent $\mathsf{G_i}$ of $\mathsf{S_i}$ such that $\Pr[(a_i, b_i) \in \mathsf{G_i}] \geq p_i/2$, and for each $(a, b) \in \mathsf{G_i}$, the probability that $(a, \widehat{b}) \in \mathsf{S_i}$ over a random choice of $\widehat{b}$ in B_i is also at least $p_i/2$. Hence, the probability that the outcome (a, b) of the first run of $\mathsf{A_P}$ in our algorithm is in $\mathsf{G_i}$ is at least $p_i/2$, and then for each of those outcomes, the probability over the random choice of $\widehat{b} = (\widehat{r}[i], \ldots, \widehat{r}[q_o])$ that the second run outcome $(a, \widehat{b})$ is in $\mathsf{S_i}$ is at least $p_i/2$. Since $\widehat{r}[i]$ is uniformly chosen in $\mathbb{Z}_{2^{l_H}}$, the chance that it collides with $r[i]$ is $1/2^{l_H}$, so with probability at least $p_i/2 - 1/2^{l_H}$ over $\widehat{b}$ we know that $(a, \widehat{b}) \in \mathsf{S_i}$ and also $\widehat{r}[i] \neq r[i]$. Summarizing, we have that the probability that (1) $(a, b) \in \mathsf{G_i}$ and (2) $(a, \widehat{b}) \in \mathsf{S_i}$ and (3) $\widehat{r}[i] \neq r[i]$ all occur is at least $p_i/2(p_i/2 - 1/2^{l_H})$. This latter event implies that both runs are i-successful and $\widehat{r}[i] \neq r[i]$, i.e. that event $\mathsf{S_i^*}$ occurs. Hence

$$\Pr[\mathsf{S_i^*}] \geq p_i/2(p_i/2 - 1/2^{l_H}) \text{ for all } i \in \{1, \ldots, q_o\}, \tag{9}$$

and since p_i is the probability that a run of $\mathsf{A_P}$ is i-successful, we know that $\sum_{i=1}^{q_o} p_i = \mathbf{Succ}_{\mathsf{A_P}, P[q_o, q_d]}(k)$. Assuming that $\mathbf{Succ}_{\mathsf{A_P}, P[q_o, q_d]}(k) \geq 2q_o/2^{l_H}$, we apply Lemma A.3 to (9) to get

$$\Pr[\mathsf{S^*}] = \sum_{i=1}^{q_o} \Pr[\mathsf{S_i^*}] \geq (1/q_o) \cdot \left(\mathbf{Succ}_{\mathsf{A_P}, P[q_o, q_d]}(k)/2 - q_o/2^{l_H}\right)^2, \tag{10}$$

which is the desired lower-bound on $\mathsf{A_S}$'s success probability. □

To complete the proof of the theorem, we apply to A the Stage 1 reduction (Lemma A.1) followed by the Stage 2 reduction (Lemma A.4) and obtain an algorithm $\mathsf{A_S}$ for SDH with the claimed success probability bound (4) and resources $t[S]$ and $q[S]$, as claimed. □

B Proof of Theorem 4.2

The proof is straightforward but we give the details for completeness. To show the perfect unconditional privacy, assuming the *direct* verifier key-reg. protocol is used, we show how to use construct the forgery strategy $\widehat{\mathsf{A_1}}$ which, for any given privacy attacker pair $(\mathsf{A_1}, \mathsf{A_2})$, will perfectly simulate the DV signature answers to $\mathsf{A_1}$'s designation queries $y_{3,i}$ without the message m^* being signed by the signer, using the corresponding secret key $x_{3,i}$ that $\mathsf{A_1}$ registered with $y_{3,i}$ during a previous key-reg. query. This shows that the convincing measure $C_{\widehat{\mathsf{A_1}}}(\mathsf{A_1}, \mathsf{A_2})$ is zero for any $(\mathsf{A_1}, \mathsf{A_2})$, as required.

Game yes. We recall first the original attack Game yes in which $\mathsf{A_1}$ and $\mathsf{A_2}$ interact.

Stage 1. The pair $(\mathsf{A_1}, \mathsf{A_2})$ is run on input $(D_G, g, y_1 = g^{x_1})$. $\mathsf{A_1}$'s oracle queries are answered as follows.

(1) $\mathsf{S}(x_1, .)$ **Queries**. When $\mathsf{A_1}$ makes ith S-query m_i, it is answered with $\sigma_i = \mathsf{S}(x_1, m_i)$.
(2) KRA **Queries**. When $\mathsf{A_1}$ makes ith key-reg. query $(x_{3,i}, y_{3,i})$ it is answered Acc if $g^{x_{3,i}} = y_{3,i}$ and Rej else.
(3) $\mathsf{A_2}$ **Queries**. When $\mathsf{A_1}$ sends message m_i to $\mathsf{A_2}$, $\mathsf{A_2}$ responds with an answer a_i.

End of Stage 1. A_1 outputs a challenge message m^* which is given to A_2. A PV signature $\sigma_i = (r^*, s^*) = \mathsf{S}(x_1, m^*)$ is generated, where $r^* = H(m^*, u^*)$ for $u^* = g^{k^*}$ for uniformly random $k^* \in \mathbb{Z}_q$, and $s^* = k^* + r^* \cdot x_1 \bmod q$. Stage 2 begins.

Stage 2. A_1 continues to make S,KRA and A_2 queries as in Stage 1 but can also make designation queries which are answered as follows:

(1) CDV **Queries**. When A_1 makes ith CDV-query $y_{3,i}$, it is answered with DV signature $\widehat{\sigma}_i = \mathsf{CDV}(y_1, y_{3,i}, m^*, \sigma^*) = (\widehat{u}_i, \widehat{K}_i)$, where $\widehat{u}_i = g^{s^*} \cdot y_1^{-r^*}$ and $\widehat{K}_i = y_{3,i}^{s^*}$.

End of Stage 2. A_2 outputs a decision $d \in \{\mathsf{yes}, \mathsf{no}\}$.

Game no. We now describe the other game where $\widehat{\mathsf{A}_1}$ interacts with A_2.

Stage 1. The pair $(\widehat{\mathsf{A}_1}, \mathsf{A}_2)$ is run on input $(D_G, g, y_1 = g^{x_1})$, where $\widehat{\mathsf{A}_1}$ is also given the program for A_1 as input. $\widehat{\mathsf{A}_1}$ now runs A_1 on same input and answers A_1's oracle queries as follows.

(1) $\mathsf{S}(x_1, .)$ **Queries**. When A_1 makes ith S-query m_i, $\widehat{\mathsf{A}_1}$ forwards it to S oracle and forwards response $\sigma_i = \mathsf{S}(x_1, m_i)$ back to A_1.

(2) KRA **Queries**. When A_1 makes ith key-reg. query $(x_{3,i}, y_{3,i})$ $\widehat{\mathsf{A}_1}$ answers Acc if $g^{x_{3,i}} = y_{3,i}$ and stores $(x_{3,i}, y_{3,i}, Acc)$ in a table T, else it answers Rej.

(3) A_2 **Queries**. When A_1 outputs a message m_i for A_2, $\widehat{\mathsf{A}_1}$ forwards it to A_2 and forwards the answer a_i back to A_1.

End of Stage 1. A_1 outputs a challenge message m^* , which is also output by $\widehat{\mathsf{A}_1}$ and given to A_2. $\widehat{\mathsf{A}_1}$ computes $u^* = g^{k^*}$ and $r^* = H(m^*, u^*)$, for a uniformly random $k^* \in \mathbb{Z}_q$. Stage 2 begins.

Stage 2. A_1 continues to make S,KRA and A_2 queries, answered by $\widehat{\mathsf{A}_1}$ as in Stage 1, but can also make designation queries which are answered as follows:

(1) CDV **Queries**. When A_1 makes ith CDV-query $y_{3,i}$, $\widehat{\mathsf{A}_1}$ searches table T for an entry $(x_{3,j}, y_{3,j}, Acc)$ with $y_{3,j} = y_{3,i}$ (note that this entry is guaranteed to exist in T due to the restriction on A_1 to only query CDV with public keys which have been answered with Acc by a previous KRA query) and answers with $\widehat{\sigma}_i = (\widehat{u}_i, \widehat{K}_i)$, where $\widehat{u}_i = u^*$ and $\widehat{K}_i = (u^* \cdot y^{-r^*})^{x_{3,j}}$.

End of Stage 2. A_2 outputs a decision $d \in \{\mathsf{yes}, \mathsf{no}\}$.

Note that A_1's (and hence also A_2) view is perfectly simulated in Game no as in Game yes. This is because in both games u^* and r^* are computed identically and the DV signatures (u_i, K_i) are also identical in both games because $\widehat{u}_i = g^{s^*} \cdot y_1^{-r^*} = u^*$ in Game yes and $u_i = u^*$ in Game no for all i, and $\widehat{K}_i = y_{3,i}^{s^*} = g^{x_{3,i}s^*} = (u^* y^{-r^*})^{x_{3,i}}$ in Game yes and $\widehat{K}_i = (u^* \cdot y^{-r^*})^{x_{3,j}} = (u^* \cdot y^{-r^*})^{x_{3,i}}$ in Game no. All other oracles are simulated identically in both games. So A_2 outputs yes with same probability in both games and hence $C_{\widehat{\mathsf{A}_1}}(\mathsf{A}_1, \mathsf{A}_2) = 0$, as claimed. The run-time of $\widehat{\mathsf{A}_1}$ is the run-time of A_1 plus the time $q_{d1} O(l_q T_g + q_{d1} l_G) + q_{k1} O(l_q T_g)$ to answer A_1's CDV and KRA queries. Note that $\widehat{q_{s1}} = q_{s1}$, $\widehat{q_{c1}} = q_{c1}$, and $\widehat{q_{k1}} = 0$, as claimed. □

C Proof of Theorem 4.3

We show how to use any efficient forging attacker A for breaking scheme $\mathsf{SchUDVS}_2$ in the sense of ST-UF-DV with non-negligible probability to construct (1) an efficient attacker A_S for breaking the PV unforgeability of scheme $\mathsf{SchUDVS}_2$ (i.e. the unforgeability of the Schnorr signature scheme), and (2) an efficient attacker A_T for breaking the collision-resistance of the trapdoor hash scheme TH, such that at least one of A_S or A_T succeed with non-negligible probability. More precisely, we show that:

$$\mathbf{Succ}^{\mathrm{UF-PV}}_{\mathsf{A}_\mathsf{S},\mathsf{SchUDVS}_2}(k) + \mathbf{Succ}^{\mathrm{CR}}_{\mathsf{A}_\mathsf{T},\mathsf{TH}}(k) \geq \frac{1}{4(q_J + q_v)q_s} \cdot \left[\mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\mathsf{A},\mathsf{SchUDVS}_2}(k) - \frac{2(q_J + q_v)q_s + 1}{2^{l_J}}\right]^2, \quad (11)$$

where $\mathsf{A_S}$ and $\mathsf{A_T}$ have resources $(t[S], q_s[S], q_H[S])$ and $(t[T])$ respectively, as defined in the theorem statement. The theorem then follows immediately from (11), by taking maximums over all attackers $\mathsf{A_S}$ with the given running time. It remains to construct $\mathsf{A_S}$ and $\mathsf{A_T}$ and show (11).

Modified Attacker $\widehat{\mathsf{A}}$. Similar to the proof of Theorem 4.1, we first define a *modified* attacker $\widehat{\mathsf{A}}$ which is obtained from the original attacker A in order to satisfy two properties (which may not be satisfied by A): (1) Each J-query $(m_i, r_i, u_i, \widehat{h}_i)$ of $\widehat{\mathsf{A}}$ is 'new' (i.e. unequal any earlier query to $J(.)$ made by $\widehat{\mathsf{A}}$), and (2) $\widehat{\mathsf{A}}$ does not make any VDV queries. Since the VDV oracle of A can be simulated knowing only the verifier's public key pk_3, and using one query to $H(.)$ and $J(.)$ per VDV query, we can easily transform A into $\widehat{\mathsf{A}}$ such that $\widehat{\mathsf{A}}$'s resources (denoted with hats) are related to A's resources as follows: $\widehat{q_s} = q_s$, $\widehat{q_v} = 0$, $\widehat{q_H} = q_H + q_v$, $\widehat{q_J} = q_J + q_v$, and $\widehat{t} = t + O((q_J + q_v)\log_2(q_J + q_v)\cdot(\ell + l_G + l_F)) + O(l_q T_g q_v)$.

$\mathsf{Game_0}$. Let $\mathsf{Game_0}$ denote the original forgery attack game. In this game, let $(m_i, r_i, u_i, \widehat{h}_i)$ denote the ith $J(.)$ query of $\widehat{\mathsf{A}}$ and $\widehat{r}_i$ the corresponding answer, let $(m^*, u^*, \widehat{r}^*_F, \widehat{r}^*, \widehat{s}^*)$ denote the output forgery of $\widehat{\mathsf{A}}$, and let m'_i denote the ith S-query of $\widehat{\mathsf{A}}$. Let $\mathsf{S_0}$ denote the event in $\mathsf{Game_0}$ that $\widehat{\mathsf{A}}$ breaks $\mathsf{SchUDVS_2}$ in the sense of ST-UF-DV. By definition, this means:

$$\mathsf{S_0} \;\Rightarrow\; \begin{array}{l}\text{(a) } J(m^*, r^*, u^*, \widehat{h}^*) = \widehat{r}^*, \text{ where } r^* = H(m^*, u^*),\ \widehat{h}^* = F_{pk}(\widehat{u}^*; \widehat{r}^*_F),\ \widehat{u}^* = g^{\widehat{s}^*}(u^* y_1^{r^*})^{-\widehat{r}^*} \\ \text{(b) } m^* \neq m'_i \text{ for all } i \in W^S,\end{array} \tag{12}$$

where W^S denotes the set $\{1, \ldots, q_s\}$ of S-query indices.

Let $\mathsf{Bad_0}$ denote the event in $\mathsf{Game_0}$ that $\widehat{\mathsf{A}}$ did not make a J-query $(m^*, r^*, u^*, \widehat{h}^*)$ during the attack.

$$\mathsf{Bad_0} \;\Rightarrow\; (m_i, r_i, u_i, \widehat{h}_i) \neq (m^*, r^*, u^*, \widehat{h}^*) \text{ for all } i \in W^J, \tag{13}$$

where W^J denotes the set $\{1, ..., q_J\}$ of J-query indices.

Thanks to the randomness of $J(.)$, we get:

Claim C.1.

$$\Pr[\mathsf{S_0} \wedge \mathsf{Bad_0}] \leq 1/2^{l_J}.$$

Proof of Claim. If $\mathsf{Bad_0}$ occurs, we know $(m^*, r^*, u^*, \widehat{h}^*)$ is not queried to $J(.)$ during the game, and therefore $\widehat{\mathsf{A}}$'s view (and hence its output $(m^*, u^*, \widehat{r}^*_F, \widehat{r}^*, \widehat{s}^*)$) is independent of $J(m^*, r^*, u^*, \widehat{h}^*)$, which is in turn uniformly distributed in $\mathbb{Z}_{2^{l_J}}$. Hence if $\mathsf{Bad_0}$ occurs the chance that $\widehat{r}^* = J(m^*, r^*, u^*, \widehat{h}^*)$ (so $\mathsf{S_0}$ also occurs) is $1/2^{l_J}$. The claim follows. □

Let $\mathsf{S_0^1}$ denote the event in $\mathsf{Game_0}$ that $\mathsf{S_0}$ occurs but $\mathsf{Bad_0}$ does not. That is:

$$\mathsf{S_0^1} \;\Rightarrow\; \begin{array}{l}\text{(a) There exists } i^* \in W^J \text{ such that } (m^*, r^*, u^*) = (m_{i^*}, r_{i^*}, u_{i^*}) \\ \quad\text{and } F_{pk}(g^{\widehat{s}^*}\cdot(u_{i^*} y_1^{r_{i^*}})^{-\widehat{r}_{i^*}}; \widehat{r}^*_F) = \widehat{h}_{i^*} \text{ and } r_{i^*} = H(m_{i^*}, u_{i^*}) \\ \text{(b) } m^* \neq m'_i \text{ for all } i \in W^S\end{array} \tag{14}$$

The above results immediately give:

Claim C.2.

$$\Pr[\mathsf{S_0^1}] \geq \Pr[\mathsf{S_0}] - \frac{1}{2^{l_J}}.$$

Proof of Claim. We have $\Pr[\mathsf{S_0^1}] = \Pr[\mathsf{S_0} \wedge \neg\mathsf{Bad_0}] \geq \Pr[\mathsf{S_0}] - \Pr[\mathsf{Bad_0}] \geq \Pr[\mathsf{S_0}] - 1/2^{l_J}$ using Claim C.1. □

Game_1. In Game_1, we construct the algorithm $\mathsf{A_S}$ against the PV-unforgeability of $\mathsf{SchUDVS}_2$. On input (k, D_G, g, y_1) $\mathsf{A_S}$ runs as follows.

Setup. $\mathsf{A_S}$ first sets up two random vectors $\overrightarrow{r}[1] = (\widehat{r}_1[1], \ldots, \widehat{r}_{\widehat{q}_J}[1])$ and $\overrightarrow{r}[2] = (\widehat{r}_1[2], \ldots, \widehat{r}_{\widehat{q}_J}[2])$ with $\widehat{r}_i[1]$'s and $\widehat{r}_i[2]$'s chosen uniformly and independently at random from $\mathbb{Z}_{2^{l_J}}$ (these vectors will be used to answer $\mathsf{A_P}$'s $J(.)$ queries). $\mathsf{A_S}$ also generates a TH key-pair $(sk, pk) = \mathsf{GKF}(k)$.

First Run. $\mathsf{A_S}$ runs $\widehat{\mathsf{A}}$ on input $(D_G, g, y_1, pk; \omega)$, where ω is a random bit string used as the randomness input of $\widehat{\mathsf{A}}$, and answers $\widehat{\mathsf{A}}$'s oracle queries as follows:

(1) $J(.)$-**Query simulator** $\mathsf{F^J}$. When $\widehat{\mathsf{A}}$ makes its ith $J(.)$ query $(m_i[1], r_i[1], u_i[1], \widehat{h}_i[1])$, $\mathsf{A_S}$ responds with $\widehat{r}_i[1]$.

(2) S-**Query simulator** $\mathsf{F^S}$. When $\widehat{\mathsf{A}}$ makes its jth S query $m'_j[1]$, $\mathsf{A_S}$ simply forwards the query to its S oracle and forwards the oracle's response $\sigma_j[1]$ back to $\widehat{\mathsf{A}}$. $\mathsf{A_S}$ stores the query-answer pair $(m'_j[1], \sigma_j[1])$ in a table T.

First Run Output. At the end of first run, $\widehat{\mathsf{A}}$ outputs the forgery $(m^*[1], u^*[1], \widehat{r}^*_F[1], \widehat{r}^*[1], \widehat{s}^*[1])$. Note that if this run is successful then there exists $i^* \in W^J$ such that $\widehat{\mathsf{A}}$'s forgery satisfies (14). We also define $j^* \in W^S$ as the number of S-queries made by $\widehat{\mathsf{A}}$ before issuing its i^*th J-query. $\mathsf{A_S}$ finds (i^*, j^*) from a table of $\widehat{\mathsf{A}}$'s queries in time $O(\widehat{q}_J(\ell + l_F + l_G) + l_q T_g)$ (if i^* doesn't exist, $\mathsf{A_S}$ fails).

Second Run. $\mathsf{A_S}$ runs $\widehat{\mathsf{A}}$ again on the *same* input $(D_g, g, y_1, pk; \omega)$ as used in first run, but answers its oracle queries differently as follows:

(1) $J(.)$-**Query simulator** $\mathsf{F^J}$. When $\widehat{\mathsf{A}}$ makes its ith $J(.)$ query $(m_i[2], r_i[2], u_i[2], \widehat{h}_i[2])$, $\mathsf{A_S}$ responds with $\widehat{r}_i[1]$ for $i < i^*$ and with $\widehat{r}_i[2]$ for $i \geq i^*$.

(2) S-**Query simulator** $\mathsf{F^S}$. When $\widehat{\mathsf{A}}$ makes its jth S query $m'_j[2]$, $\mathsf{A_S}$ responds with $\sigma_j[1]$ for $j \leq j^*$. For $j > j^*$, $\mathsf{A_S}$ forwards the query to its S oracle and forwards the oracle's response $\sigma_j[2]$ back to $\widehat{\mathsf{A}}$.

Second Run Output. At the end of second run, $\widehat{\mathsf{A}}$ outputs the forgery $(m^*[2], u^*[2], \widehat{r}^*_F[2], \widehat{r}^*[2], \widehat{s}^*[2])$.

$\mathsf{A_S}$*'s output.* $\mathsf{A_S}$ computes and returns an estimate $(\widehat{m}^*, (\widehat{r}^*, \widehat{s}^*))$ for a message/PV sig. forgery for $\mathsf{SchUDVS}_2$, where $\widehat{m}^* = m^*[1]$, $\widehat{r}^* = H(m^*[1], u^*[1])$ and $\widehat{s}^* = (\widehat{s}^*[1] - \widehat{s}^*[2]) \cdot (\widehat{r}_{i^*}[2] - \widehat{r}_{i^*}[1])^{-1} \bmod q$ (if $\widehat{r}_{i^*}[1] = \widehat{r}_{i^*}[2]$ then $\mathsf{A_S}$ fails).

This completes the description of $\mathsf{A_S}$. The running-time of $\mathsf{A_S}$ is twice the run-time of $\widehat{\mathsf{A}}$ plus the time to compute (i^*, j^*) and $\widehat{s}^*$ at the end, which takes total time $O(\widehat{q}_J(\ell + l_F + l_G) + l_q T_g + l_q^2)$. The number of H- and S- queries made by $\mathsf{A_S}$ is up to twice the number of queries made by $\widehat{\mathsf{A}}$. This establishes the claimed resources of $\mathsf{A_S}$.

The collision-finder attacker $\mathsf{A_T}$ runs $\widehat{\mathsf{A}}$ twice in the same way as $\mathsf{A_S}$, and at the end computes the following collision estimate $(\alpha[1], \beta[1]), (\alpha[2], \beta[2])$, where $\alpha[\rho] = (u_{i^*}[1] \cdot y_1^{r_{i^*}[1]})^{-\widehat{r}_{i^*}[\rho]}$ and $\beta[\rho] = \widehat{r}_F[\rho]$ for $\rho \in \{1, 2\}$.

We now lower bound the sum of success probabilities of $\mathsf{A_S}$ and $\mathsf{A_T}$. For each $(i, j) \in W^J \times W^S$, we call a run of $\widehat{\mathsf{A}}$ (i, j)-successful if $\widehat{\mathsf{A}}$'s output satisfies (14) and $(i^*, j^*) = (i, j)$. Let S^* denote the event that both runs of $\widehat{\mathsf{A}}$ above are (i, j)-successful for some (i, j) and $\widehat{r}_i[1] \neq \widehat{r}_i[2]$. Note that if S^* occurs then, defining s^* as the discrete-log $u_i[1] \cdot y_1^{r_i[1]}$ to base g in G, we have from (14) that

$$\widehat{m}^* = m^*[1] = m_i[1] = m^*[2] = m_i[2] \text{ has not been queried to } \mathsf{S} \text{ during either run of } \widehat{\mathsf{A}} \qquad (15)$$

and

$$F_{pk}(g^{\widehat{s}_i[1]} \cdot (g^{s^*})^{-\widehat{r}_i[1]}; \widehat{r}^*_F[1]) = F_{pk}(g^{\widehat{s}_i[1]} \cdot (g^{s^*})^{-\widehat{r}_i[2]}; \widehat{r}^*_F[2]) \qquad (16)$$

(note that $(m_i[1], r_i[1], u_i[1]) = (m_i[2], r_i[2], u_i[2])$ because the view of $\widehat{\mathsf{A}}$ is the same in both runs up to ith $J(.)$ query response). We now split event S^* into two disjoint subevents $\mathsf{S}^*_{\mathsf{S}}$ and $\mathsf{S}^*_{\mathsf{T}}$, depending on whether

$$g^{\widehat{s}_i[1]} \cdot (g^{s^*})^{-\widehat{r}_i[1]} = g^{\widehat{s}_i[1]} \cdot (g^{s^*})^{-\widehat{r}_i[2]} \tag{17}$$

holds or not, respectively. If the subevent $\mathsf{S}^*_{\mathsf{S}}$ occurs, then (17) holds and hence A_{S}'s estimate $\widehat{s}^*$ is equal to the discrete-log $s^* = (\widehat{s}^*[1] - \widehat{s}^*[2]) \cdot (\widehat{r}_{i^*}[2] - \widehat{r}_{i^*}[1])^{-1} \bmod q$ of $u_i[1] y_1^{r_i[1]}$ and $r_i[1] = H(m_i[1], u_i[1])$. Thus $\mathsf{S}^*_{\mathsf{S}}$ implies that A_{S}'s forgery is valid and coupled with (15) means that A_{S} succeeds to break the PV unforgeability of $\mathsf{SchUDVS}_2$ in this case. In the other case that subevent $\mathsf{S}^*_{\mathsf{T}}$ occurs, (17) does not hold but (16) holds, meaning that A_{T}'s output is a valid collision for the trapdoor hash so A_{T} succeeds.

So we have shown that the sum of success probabilities of A_{S} and A_{T} is equal to the probability of the event S^* that both runs of $\widehat{\mathsf{A}}$ are (i,j)-successful for some $(i,j) \in W^J \times W^S$ and $\widehat{r}_i[1] \neq \widehat{r}_i[2]$, and it remains to lower bound $\Pr[\mathsf{S}^*]$. To do this, we split S^* into $|W_J \times W^S| = \widehat{q}_J q_s$ disjoint subevents ${\mathsf{S}^*}_{i,j}$ according the value of (i,j) and bound each one. For each (i,j), let $A_{i,j}$ denote the outcome space for the random variable $a_{i,j} = (D_g, g, y_1, pk, \omega, (r_1, \ldots, r_{i-1}), (k_1, \ldots, k_j))$ consisting of the view of $\widehat{\mathsf{A}}$ up to the ith query to $O(.)$ and the random elements k_i used by S oracle to answer first j signature queries, and let $B_{i,j}$ denote the outcome space for the independent random variable $b_{i,j} = ((r_i, \ldots, r_{\widehat{q}_J}), (k_{j+1}, \ldots, k_{q_s}))$ consisting of the view of A^{P} after the ith query to $O(.)$ (including the response $r[i]$ to the ith query) and k_i's used to answer all remaining signature queries . Note that the event $\mathsf{S}_{i,j}$ that a run of $\widehat{\mathsf{A}}$ is (i,j)-successful is a subset of $A_{i,j} \times B_{i,j}$ with probability $p_{i,j} \overset{\text{def}}{=} \Pr[(a_{i,j}, b_{i,j}) \in \mathsf{S}_{i,j}]$. Proceeding from this point analgously to the calculation in Lemma A.4, we apply the Splitting Lemma A.2 and obtain

$$\Pr[{\mathsf{S}^*}_{i,j}] \geq p_{i,j}/2(p_{i,j}/2 - 1/2^{l_J}) \text{ for all } (i,j) \in W^J \times W^S, \tag{18}$$

and hence applying Lemma A.3 to (18) noting that $\sum_{i,j} p_{i,j} = \Pr[\mathsf{S}^1_0]$ we get

$$\Pr[\mathsf{S}^*] = \sum_{(i,j) \in W^J \times W^S} \Pr[{\mathsf{S}^*}_{i,j}] \geq \frac{1}{\widehat{q}_J q_s} \cdot \left(\Pr[\mathsf{S}^1_0]/2 - \widehat{q}_J q_s / 2^{l_J}\right)^2, \tag{19}$$

which using Claim C.2 gives the desired lower-bound (11) on $\Pr[\mathsf{S}^*]$. This completes the proof. $\square$

D Proof of Theorem 4.4

To show the perfect unconditional privacy, assuming the *direct* verifier key-reg. protocol is used, we show how to construct the forgery strategy $\widehat{\mathsf{A}_1}$ which, for any given privacy attacker pair $(\mathsf{A}_1, \mathsf{A}_2)$, will perfectly simulate the DV signature answers to A_1's designation queries pk_i without the message m^* being signed by the signer, using the corresponding secret key sk_i that A_1 registered with pk_i during a previous key-reg. query. This shows that the convincing measure $C_{\widehat{\mathsf{A}_1}}(\mathsf{A}_1, \mathsf{A}_2)$ is zero for any $(\mathsf{A}_1, \mathsf{A}_2)$, as required.

Game yes. We recall first the original attack Game yes in which A_1 and A_2 interact.

Stage 1. The pair $(\mathsf{A}_1, \mathsf{A}_2)$ is run on input $(D_G, g, y_1 = g^{x_1})$. A_1's oracle queries are answered as follows.

(1) $\mathsf{S}(x_1, .)$ **Queries**. When A_1 makes ith S-query m_i, it is answered with $\sigma_i = \mathsf{S}(x_1, m_i)$.

(2) KRA **Queries**. When A_1 makes ith key-reg. query (r_i, sk_i, pk_i) it is answered Acc if $(sk_i, pk_i) = \mathsf{GKF}(k; r_i)$ and Rej else.

(3) A_2 **Queries**. When A_1 sends message m_i to A_2, A_2 responds with an answer a_i.

End of Stage 1. A_1 outputs a challenge message m^* which is also given to A_2. A PV signature $\sigma_i = (r^*, s^*) = \mathsf{S}(x_1, m^*)$ is generated, where $r^* = H(m^*, u^*)$ for $u^* = g^{k^*}$ for uniformly random $k^* \in \mathbb{Z}_q$, and $s^* = k^* + r^* \cdot x_1 \bmod q$. Stage 2 begins.

Stage 2. A_1 continues to make S,KRA and A_2 queries as in Stage 1 but can also make designation queries which are answered as follows:

(1) CDV **Queries**. When A_1 makes ith CDV-query pk_i, it is answered with DV signature $\widehat{\sigma}_i = \mathsf{CDV}(y_1, pk_i, m^*, \sigma^*) = (u_i, \widehat{r}_{F,i}, \widehat{r}_i, \widehat{s}_i)$, where $\widehat{u}_i = g^{s^*} \cdot y_1^{-r^*}$, $\widehat{r}_i = J(m^*, r^*, u^*, \widehat{h}_i)$ and $\widehat{s}_i = \widehat{k}_i + \widehat{r}_i \cdot s^* \bmod q$, where $\widehat{h}_i = F_{pk}(\widehat{u}_i; \widehat{r}_{F,i})$ and $\widehat{u}_i = g^{\widehat{k}_i}$ for uniformly random $\widehat{k}_i \in \mathbb{Z}_q$.

End of Stage 2. A_2 outputs a decision $d \in \{\mathsf{yes}, \mathsf{no}\}$.

Game no. We now describe the other game where $\widehat{\mathsf{A}_1}$ interacts with A_2.

Stage 1. The pair $(\widehat{\mathsf{A}_1}, \mathsf{A}_2)$ is run on input $(D_G, g, y_1 = g^{x_1})$, , where $\widehat{\mathsf{A}_1}$ is also given the program for A_1 as input. $\widehat{\mathsf{A}_1}$ runs A_1 on same input and answers A_1's oracle queries as follows.

(1) $\mathsf{S}(x_1,.)$ **Queries**. When A_1 makes ith S-query m_i, $\widehat{\mathsf{A}_1}$ forwards it to S oracle and forwards response $\sigma_i = \mathsf{S}(x_1, m_i)$ back to A_1.

(2) KRA **Queries**. When A_1 makes ith key-reg. query (r_i, sk_i, pk_i), $\widehat{\mathsf{A}_1}$ answers Acc if $(sk_i, pk_i) = \mathsf{GKF}(k; r_i)$ and stores (sk_i, pk_i, Acc) in a table T, else it answers Rej.

(3) A_2 **Queries**. When A_1 outputs a message m_i for A_2, $\widehat{\mathsf{A}_1}$ forwards it to A_2 and forwards the answer a_i back to A_1.

End of Stage 1. A_1 outputs a challenge message m^* , which is also output by $\widehat{\mathsf{A}_1}$ and given to A_2. $\widehat{\mathsf{A}_1}$ computes $u^* = g^{k^*}$ and $r^* = H(m^*, u^*)$, for a uniformly random independent $k^* \in \mathbb{Z}_q$. Stage 2 begins.

Stage 2. A_1 continues to make S,KRA and A_2 queries, answered by $\widehat{\mathsf{A}_1}$ as in Stage 1, but can also make designation queries which are answered as follows:

(1) CDV **Queries**. When A_1 makes ith CDV-query pk_i, $\widehat{\mathsf{A}_1}$ searches table T for an entry (sk_j, pk_j, Acc) with $pk_j = pk_i$ (note that this entry is guaranteed to exist in T due to the restriction on A_1 to only query CDV with public keys which have been answered with Acc by a previous KRA query) and answers with $\widehat{\sigma}_i = (u_i, \widehat{r}_{F,i}, \widehat{r}_i, \widehat{s}_i)$, where $\widehat{u}_i = u^*$, $\widehat{r}_i = J(m^*, r^*, u^*, \widehat{h}_i)$, $\widehat{h}_i = F_{pk}(\widehat{u}'; \widehat{r}'_{F,i})$ for some fixed $\widehat{u}' \in G$ and uniformly random and independent $\widehat{r}'_{F,i} \in R_F$, $\widehat{s}_i$ is uniformly random and independent in $\mathbb{Z}_q$, and $\widehat{r}_{F,i} = \mathsf{CSF}((sk_j, pk_j), (\widehat{u}', \widehat{r}'_{F,i}), \widehat{u}_i)$, with $\widehat{u}_i = g^{\widehat{s}_i}(u^* y_1^{r^*})^{-\widehat{r}_i}$.

End of Stage 2. A_2 outputs a decision $d \in \{\mathsf{yes}, \mathsf{no}\}$.

We show that A_1's (and hence also A_2) view is perfectly simulated in Game no as in Game yes. In particular, the DV signatures $(\widehat{\sigma}_i = (u_i, \widehat{r}_{F,i}, \widehat{r}_i, \widehat{s}_i)$ are distributed identically in both games, for the following reasons. First, note that u^* and r^* are computed identically in both games. Second, observe that for each DV signature, $\widehat{r}_i$ and $\widehat{s}_i$ are determined uniquely by $(m^*, r^*, u^*, \widehat{u}_i, \widehat{r}_{F,i})$ in the same way in both games, namely

$$\widehat{r}_i = J(m^*, r^*, u^*, F_{pk}(\widehat{u}_i; \widehat{r}_{F,i}))$$

and

$$\widehat{s}_i \text{ is the discrete-log in } G \text{ of } \widehat{u}_i(u^* y_1^{r^*})^{\widehat{r}_i} \text{ to base } g$$

in both games. So it remains to show that (for each DV signature) the pair $(\widehat{u}_i, \widehat{r}_{F,i})$ is identically distributed in both games. In Game yes, $(\widehat{u}_i, \widehat{r}_{F,i})$ is uniform on $G \times R_F$ by definition. In Game no, we have that $(\widehat{u}_i, \widehat{r}_{F,i}) = G(\widehat{s}_i, \widehat{r}'_{F,i})$ for a function $G : \mathbb{Z}_q \times R_F \to G \times R_F$ defined by:

$$\widehat{u}_i = g^{\widehat{s}_i}(u^* y_1^{r^*})^{-J(m^*, r^*, u^*, F_{pk}(\widehat{u}'; \widehat{r}'_{F,i}))} \quad (20)$$

$$\widehat{r}_{F,i} = \mathsf{CSF}((sk, pk), (\widehat{u}', \widehat{r}'_{F,i}), \widehat{u}_i). \quad (21)$$

From the perfectly-trapdoor property of TH we have that mapping $\widehat{r}'_{F,i} \mapsto \mathsf{CSF}((sk, pk), (\widehat{u}', \widehat{r}'_{F,i}), \widehat{u}_i)$ is a permutation on R_F. This and the fact that g has order q immediately implies that function $G : \mathbb{Z}_q \times R_F \to G \times R_F$ is one-to-one. So since $(\widehat{s}_i, \widehat{r}'_{F,i})$ is uniform on $\mathbb{Z}_q \times R_F$ by definition, it follows that the image pair $(\widehat{u}_i, \widehat{r}_{F,i})$ is uniform on $G \times R_F$ in Game no, as required. We conclude that A_2 outputs yes with same probability in both games and hence $C_{\widehat{\mathsf{A}_1}}(\mathsf{A}_1, \mathsf{A}_2) = 0$, as claimed. The run-time of $\widehat{\mathsf{A}_1}$ is the run-time of A_1 plus the time $q_{d1} O(l_q T_g + T_F + T_{CSF} + T_J + T_H + q_{d1} l_{pk}) + q_{k1} T_{GKF}$ to answer A_1's CDV and KRA queries. Note that $\widehat{q_{s1}} = q_{s1}$, $\widehat{q_{c1}} = q_{c1}$, and $\widehat{q_{k1}} = 0$, as claimed. □

E Proof of Theorem 5.1

The proof is analogous to the proof of Theorem 4.1 so we don't provide all details. We show how to use any efficient forging attacker A for breaking scheme $\mathsf{RSAUDVS}$ in the sense of ST-UF-DV with non-negligible probability to construct (1) an efficient attacker $\mathsf{A_S}$ for breaking the PV unforgeability of scheme $\mathsf{RSAUDVS}$ (i.e. the unforgeability of the standard RSA signature scheme), and (2) an efficient attacker $\mathsf{A_T}$ for breaking the collision-resistance of the trapdoor hash scheme TH, such that at least one of $\mathsf{A_S}$ or $\mathsf{A_T}$ succeed with non-negligible probability. More precisely, we show that:

$$\mathbf{Succ}^{\mathrm{UF-PV}}_{\mathsf{A_S},\mathsf{RSAUDVS}}(k) + \mathbf{Succ}^{\mathrm{CR}}_{\mathsf{A_T},\mathsf{TH}}(k) \geq \frac{1}{4(q_J + q_v)q_s} \cdot \left[\mathbf{Succ}^{\mathrm{ST-UF-DV}}_{\mathsf{A},\mathsf{RSAUDVS}}(k) - \frac{2(q_J + q_v)q_s + 1}{2^{l_J}}\right]^2, \quad (22)$$

where $\mathsf{A_S}$ and $\mathsf{A_T}$ have resources $(t[S], q_s[S], q_H[S])$ and $(t[T])$ respectively, as defined in the theorem statement. The theorem then follows immediately from (22), by taking maximums over all attackers $\mathsf{A_S}$ with the given running time. It remains to construct $\mathsf{A_S}$ and $\mathsf{A_T}$ and show (22).

Modified Attacker $\widehat{\mathsf{A}}$. As in Theorem 4.3, we first define a *modified* attacker $\widehat{\mathsf{A}}$ which is obtained from the original attacker A in order to satisfy two properties (which may not be satisfied by A): (1) Each J-query of $\widehat{\mathsf{A}}$ is 'new' (i.e. unequal any earlier query to $J(.)$ made by $\widehat{\mathsf{A}}$), and (2) $\widehat{\mathsf{A}}$ does not make any VDV queries. The $\widehat{\mathsf{A}}$'s resources (denoted with hats) are related to A's resources as follows: $\widehat{q}_s = q_s$, $\widehat{q}_v = 0$, $\widehat{q}_H = q_H + q_v$, $\widehat{q}_J = q_J + q_v$, and $\widehat{t} = t + O((q_J + q_v)\log_2(q_J + q_v) \cdot (\ell + l_G + l_F)) + O(l_q T_g q_v)$.

$\mathsf{Game_0}$. Let $\mathsf{Game_0}$ denote the original forgery attack game. In this game, let $(m_i, h_i, \widehat{h}_i)$ denote $\widehat{\mathsf{A}}$'s ith J-query, and $\widehat{r}_i = (\widehat{r}_{i,1}, \ldots, \widehat{r}_{i,\alpha})$ the response to this query. Let $(m^*, h^*, \widehat{r}^*_F, \widehat{r}^*, \widehat{s}^*)$ denote $\widehat{\mathsf{A}}$'s output forgery, with $\widehat{r}^* = (\widehat{r}^*_1, \ldots, \widehat{r}^*_\alpha)$, $\widehat{s}^* = (\widehat{s}^*_1, \ldots, \widehat{s}^*_\alpha)$, $\widehat{h}^* = F_{pk}(\widehat{u}^*, \alpha; \widehat{r}^*_F)$, $\widehat{u}^* = (\widehat{u}^*_1, \ldots, \widehat{u}^*_\alpha)$ and $\widehat{u}^*_i = (\widehat{s}^*_i)^e \cdot (h^*)^{-\widehat{r}^*_i}$ for $i = 1, \ldots, \alpha$. Let $W_J = \{1, \ldots, \widehat{q}_J\}$ and $W_S = \{1, \ldots, \widehat{q}_s\}$, and let m'_i denote the ith S query of $\widehat{\mathsf{A}}$. Analogously to Theorem 4.3, we define the event $\mathsf{S_0}$ that A succeeds and the event $\mathsf{S_0^1}$:

$$\begin{aligned} \mathsf{S_0^1} \Rightarrow\ & \text{(a) There exists } i^* \in W^J \text{ such that } (m^*, h^*, \widehat{h}^*) = (m_i, h_{i^*}, \widehat{h}_{i^*}) \\ & \quad \text{and } F_{pk}(\widehat{u}_{i^*,1}, \ldots, \widehat{u}_{i^*,\alpha}; \widehat{r}^*_F) = \widehat{h}_{i^*} \text{ with } \widehat{u}_{i^*,l} = (\widehat{s}^*_l)^e \cdot h_{i^*}^{-\widehat{r}_{i^*,l}} \bmod N \\ & \text{(b) } m^* \neq m'_i \text{ for all } i \in W^S \text{ and } R(m^*, h^*) = Acc \end{aligned} \quad (23)$$

and due to the randomness of $J(.)$ we get

$$\Pr[\mathsf{S}_0^1] \geq \Pr[\mathsf{S}_0] - \frac{1}{2^{l_J}}. \tag{24}$$

Game_1. In Game_1, we construct the algorithm $\mathsf{A_S}$ against the PV-unforgeability of $\mathsf{RSAUDVS}_2$. On input (k, N, e) $\mathsf{A_S}$ runs as follows.

Setup. $\mathsf{A_S}$ first sets up two random vectors $\overrightarrow{\widehat{r}}[1] = (\widehat{r}_1[1], \ldots, \widehat{r}_{\widehat{q}_J}[1])$ and $\overrightarrow{\widehat{r}}[2] = (\widehat{r}_1[2], \ldots, \widehat{r}_{\widehat{q}_J}[2])$ with $\widehat{r}_i[k] = (\widehat{r}_{i,1}[k], \ldots, \widehat{r}_{i,\alpha}[k])$ chosen uniformly and independently at random from $\mathbb{Z}_{2^{l_J/\alpha}}^{\alpha}$ for $k \in \{1, 2\}$.

First Run. $\mathsf{A_S}$ generates a TH key-pair $(sk, pk) = \mathsf{GKF}(k)$ and runs $\widehat{\mathsf{A}}$ on input $(N, e, pk; \omega)$, where ω is a random bit string used as the randomness input of $\widehat{\mathsf{A}}$, and answers $\widehat{\mathsf{A}}$'s oracle queries as follows:

(1) $J(.)$**-Query simulator** $\mathsf{F^J}$. When $\widehat{\mathsf{A}}$ makes its ith $J(.)$ query $(m_i[1], h_i[1], \widehat{h}_i[1])$, $\mathsf{A_S}$ responds with $\widehat{r}_i[1]$.

(2) S-**Query simulator** $\mathsf{F^S}$. When $\widehat{\mathsf{A}}$ makes its jth S query $m'_j[1]$, $\mathsf{A_S}$ simply forwards the query to its S oracle and forwards the oracle's response $\sigma_j[1]$ back to $\widehat{\mathsf{A}}$. $\mathsf{A_S}$ stores the query-answer pair $(m'_j[1], \sigma_j[1])$ in a table T.

First Run Output. At the end of first run, $\widehat{\mathsf{A}}$ outputs the forgery $(m^*[1], h^*[1], \widehat{r}_F^*[1], \widehat{r}^*[1], \widehat{s}^*[1])$. Note that if this run is successful then there exists $i^* \in W^J$ such that $\widehat{\mathsf{A}}$'s forgery satisfies (23). We also define $j^* \in W^S$ as the number of S-queries made by $\widehat{\mathsf{A}}$ before issuing its i^*th J-query. $\mathsf{A_S}$ finds (i^*, j^*) from a table of $\widehat{\mathsf{A}}$'s queries in time $O(\widehat{q}_J(l_F + l_N))$ (if i^* doesn't exist, $\mathsf{A_S}$ fails).

Second Run. $\mathsf{A_S}$ runs $\widehat{\mathsf{A}}$ again on the *same* input $(N, e, pk; \omega)$ as used in first run, but answers its oracle queries differently as follows:

(1) $J(.)$**-Query simulator** $\mathsf{F^J}$. When $\widehat{\mathsf{A}}$ makes its ith $J(.)$ query $(m_i[2], h_i[2], \widehat{h}_i[2])$, $\mathsf{A_S}$ responds with $\widehat{r}_i[1]$ for $i < i^*$ and with $\widehat{r}_i[2]$ for $i \geq i^*$.

(2) S-**Query simulator** $\mathsf{F^S}$. When $\widehat{\mathsf{A}}$ makes its jth S query $m'_j[2]$, $\mathsf{A_S}$ responds with $\sigma_j[1]$ for $j \leq j^*$. For $j > j^*$, $\mathsf{A_S}$ forwards the query to its S oracle and forwards the oracle's response $\sigma_j[2]$ back to $\widehat{\mathsf{A}}$.

Second Run Output. At the end of second run, $\widehat{\mathsf{A}}$ outputs the forgery $(m^*[2], h^*[2], \widehat{r}_F^*[2], \widehat{r}^*[2], \widehat{s}^*[2])$.

$\mathsf{A_S}$*'s output.* $\mathsf{A_S}$ computes and returns an estimate $(\widehat{m}^*, \widehat{\sigma}^*)$ for a message/PV sig. forgery for $\mathsf{RSAUDVS}$ as follows. First, $\mathsf{A_S}$ tries to find $l^* \in \{1, \ldots, \alpha\}$ such that the integer $\delta_r = \widehat{r}_{i^*,l^*}[1] - \widehat{r}_{i^*,l^*}[2]$ is non-zero (otherwise, if $\widehat{r}_{i^*}[1] = \widehat{r}_{i^*}[2]$, then $\mathsf{A_S}$ fails). Because $\widehat{r}_{i,l^*}^*[1]$ and $\widehat{r}_{i,l^*}^*[2]$ are in $\mathbb{Z}_{2^{l_J/\alpha}}$ we know that $|\delta_r| < 2^{l_J/\alpha} < e$ and hence $\gcd(\delta_r, e) = 1$, since e is prime. So there exist integers $c_r < e$ and $c_e < e$ such that $c_r \cdot \delta_r + c_e \cdot e = 1$ and $\mathsf{A_S}$ can compute them in time $O(l_e^2)$. Then $\mathsf{A_S}$ computes the PV sig. estimate $\widehat{\sigma}^* = (\widehat{s}_{l^*}^*[1]/\widehat{s}_{l^*}^*[2])^{c_r} \cdot (h^*[1])^{c_e} \bmod N$ on message $\widehat{m}^* = m^*[1]$.

This completes the description of $\mathsf{A_S}$. The running-time of $\mathsf{A_S}$ is twice the run-time of $\widehat{\mathsf{A}}$ plus the time to compute (i^*, j^*) and $\widehat{\sigma}^*$ at the end, which takes total time $O(\widehat{q}_J(l_F + l_N) + l_e^2 + l_e T_N)$. The number of H- and S- queries made by $\mathsf{A_S}$ is up to twice the number of queries made by $\widehat{\mathsf{A}}$. This establishes the claimed resources of $\mathsf{A_S}$.

The collision-finder attacker $\mathsf{A_T}$ runs $\widehat{\mathsf{A}}$ twice in the same way as $\mathsf{A_S}$ (except that it receives the hash function public key pk as input, and generates a signature key pair $(N, e, d) = \mathsf{GKS}(k)$ by itself, with which it answers $\widehat{\mathsf{A}}$'s signing queries - note that the hash function secret key sk is not needed by $\mathsf{A_T}$), and at the end computes the following collision estimate $(\beta[1], \gamma[1]), (\beta[2], \gamma[2])$, where $\beta[\rho] = (\widehat{u}_1^*[\rho], \ldots, \widehat{u}_\alpha^*[\rho])$ and $\gamma[\rho] = \widehat{r}_F[\rho]$ for $\rho \in \{1, 2\}$ with $\widehat{u}_l^*[\rho] = \widehat{s}_l^*[\rho]^e \cdot (h_{i^*})^{-\widehat{r}_{i^*,l}[\rho]}$ for $l \in \{1, \ldots, \alpha\}$.

We now lower bound the sum of success probabilities of $\mathsf{A_S}$ and $\mathsf{A_T}$. For each $(i, j) \in W^J \times W^S$, we call a run of $\widehat{\mathsf{A}}$ (i, j)-successful if $\widehat{\mathsf{A}}$'s output satisfies (23) and $(i^*, j^*) = (i, j)$. Let S^* denote the event that both runs of $\widehat{\mathsf{A}}$ above are (i,j)-successful for some (i,j) and $\widehat{r}_i[1] \neq \widehat{r}_i[2]$. Note that if S^* occurs then, defining $\sigma^* = (h_i[1])^{1/e} \bmod N$, we have from (23) that

$$\widehat{m}^* = m_i[1] = m_i[2] \text{ has not been queried to } \mathsf{S} \text{ during either run of } \widehat{\mathsf{A}} \text{ and } R(m_i[1], h_i[1]) = Acc \quad (25)$$

and

$$F_{pk}(\widehat{u}_{i,1}[1], \ldots, \widehat{u}_{i,\alpha}[1]; \widehat{r}^*_F[1]) = F_{pk}(\widehat{u}_{i,1}[2], \ldots, \widehat{u}_{i,\alpha}[2]; \widehat{r}^*_F[2])$$
$$\text{with } \widehat{u}_{i,l}[\rho] = (\widehat{s}^*_l[\rho])^e \cdot h_i[\rho]^{-\widehat{r}_{i,l}[\rho]} \bmod N \text{ for } \rho \in \{1,2\} \text{ and } l \in \{1, \ldots, \alpha\} \quad (26)$$

(note that $(m_i[1], h_i[1], \widehat{h}_i[1]) = (m_i[2], h_i[2], \widehat{h}_i[2])$ because the view of $\widehat{\mathsf{A}}$ is the same in both runs up to ith $J(.)$ query response). We now split event S^* into two disjoint subevents $\mathsf{S}^*_{\mathsf{S}}$ and $\mathsf{S}^*_{\mathsf{T}}$, depending on whether

$$\widehat{u}_{i,l}[1] = \widehat{u}_{i,l}[2] \text{ for all } l \in \{1, \ldots, \alpha\} \quad (27)$$

holds or not, respectively. If the subevent $\mathsf{S}^*_{\mathsf{S}}$ occurs, then (27) holds. This means in particular that there exists l^* such that $\widehat{u}_{i,l^*}[1] = \widehat{u}_{i,l^*}[2]$ but $\widehat{r}_{i,l^*}[1] \neq \widehat{r}_{i,l^*}[2]$, which leads to $(\widehat{s}^*_{l^*}[1]/\widehat{s}^*_{l^*}[2]) \equiv (\sigma^*)^{\delta_r} \bmod N$ and hence $\mathsf{A_S}$'s estimate $\widehat{\sigma}^* = (\sigma^*)^{c_r \delta_r} \cdot (\sigma^*)^{c_e e} \bmod N = \sigma^* = h_i[1]^{1/e} \bmod N$, which coupled with (25) means that $R(m_i[1], h_i[1]) = Acc$ and $m_i[1]$ has not been queried to S, so $\mathsf{A_S}$ succeeds to break the PV unforgeability of $\mathsf{RSAUDVS}$ when $\mathsf{S}^*_{\mathsf{S}}$ occurs. In the other case that subevent $\mathsf{S}^*_{\mathsf{T}}$ occurs, (27) does not hold but (26) holds, meaning that $\mathsf{A_T}$'s output is a valid collision for the trapdoor hash so $\mathsf{A_T}$ succeeds.

So we have shown that the sum of success probabilities of $\mathsf{A_S}$ and $\mathsf{A_T}$ is equal to the probability of the event S^* that both runs of $\widehat{\mathsf{A}}$ are (i,j)-successful for some $(i,j) \in W^J \times W^S$ and $\widehat{r}_i[1] \neq \widehat{r}_i[2]$, and it remains to lower bound $\Pr[\mathsf{S}^*]$. Using the same calculation used to bound $\Pr[\mathsf{S}^*]$ in the proof of Theorem 4.3, we get

$$\Pr[\mathsf{S}^*] \geq \frac{1}{\widehat{q}_J q_s} \cdot \left(\Pr[\mathsf{S}^1_0]/2 - \widehat{q}_J q_s / 2^{l_J}\right)^2, \quad (28)$$

which using (24) gives the desired lower-bound (22) on $\Pr[\mathsf{S}^*]$. This completes the proof. □

F Proof of Theorem 5.2

Analogously to the proof of Theorem 4.4, we assume the *direct* verifier key-reg. protocol is used and we show how to construct the forgery strategy $\widehat{\mathsf{A}_1}$.

Game yes. We recall first the original attack Game yes in which A_1 and A_2 interact.

Stage 1. The pair $(\mathsf{A}_1, \mathsf{A}_2)$ is run on input (N, e). A_1's oracle queries are answered as follows.

(1) $\mathsf{S}(x_1, .)$ **Queries**. When A_1 makes ith S-query m_i, it is answered with $\sigma_i = \mathsf{S}(x_1, m_i)$.

(2) KRA **Queries**. When A_1 makes ith key-reg. query (r_i, sk_i, pk_i) it is answered Acc if $(sk_i, pk_i) = \mathsf{GKF}(k; r_i)$ and Rej else.

(3) A_2 **Queries**. When A_1 sends message m_i to A_2, A_2 responds with an answer a_i.

End of Stage 1. A_1 outputs a challenge message m^*, which is given to A_2. A PV signature $\sigma^* = \mathsf{S}(x_1, m^*)$ is generated, where $\sigma^* = (h^*)^{1/e} \bmod N$ and $h^* = H(m^*, s^*)$ for uniformly random $s^* \in R_S$. Stage 2 begins.

Stage 2. A_1 continues to make S,KRA and A_2 queries as in Stage 1 but can also make designation queries which are answered as follows:

(1) CDV **Queries**. When A_1 makes ith CDV-query pk_i, it is answered with DV signature $\widehat{\sigma}_i = (h_i, \widehat{r}_{F,i}, \widehat{r}_i, \widehat{s}_i)$, where $h_i = h^*$, $\widehat{r}_{F,i}$ is uniformly random in R_F, $\widehat{r}_i = (\widehat{r}_{i,1}, \dots, \widehat{r}_{i,\alpha}) = J(m^*, h^*, \widehat{h}_i)$, $\widehat{s}_i = (\widehat{s}_{i,1}, \dots, \widehat{s}_{i,\alpha})$ with $\widehat{s}_{i,l} = k_{i,l} \cdot (\sigma^*)^{\widehat{r}_{i,l}} \bmod N$, α random elements $k_i \in \mathbb{Z}_N^*$, and $\widehat{h}_i = F_{pk}(\widehat{u}_i; \widehat{r}_{F,i})$ with $\widehat{u}_i = (\widehat{u}_{i,1}, \dots, \widehat{u}_{i,\alpha})$,$\widehat{u}_{i,l} = k_{i,l}^e \bmod N$ for $l = 1, \dots, \alpha$.

End of Stage 2. A_2 outputs a decision $d \in \{\mathsf{yes}, \mathsf{no}\}$.

Game no. We now describe the other game where $\widehat{\mathsf{A}_1}$ interacts with A_2.

Stage 1. The pair $(\widehat{\mathsf{A}_1}, \mathsf{A}_2)$ is run on input (N, e), , where $\widehat{\mathsf{A}_1}$ is also given the program for A_1 as input. $\widehat{\mathsf{A}_1}$ runs A_1 on same input and answers A_1's oracle queries as follows.

(1) $\mathsf{S}(x_1, .)$ **Queries**. When A_1 makes ith S-query m_i, $\widehat{\mathsf{A}_1}$ forwards it to S oracle and forwards response $\sigma_i = \mathsf{S}(x_1, m_i)$ back to A_1.
(2) KRA **Queries**. When A_1 makes ith key-reg. query (r_i, sk_i, pk_i), $\widehat{\mathsf{A}_1}$ answers Acc if $(sk_i, pk_i) = \mathsf{GKF}(k; r_i)$ and stores (sk_i, pk_i, Acc) in a table T, else it answers Rej.
(3) A_2 **Queries**. When A_1 outputs a message m_i for A_2, $\widehat{\mathsf{A}_1}$ forwards it to A_2 and forwards the answer a_i back to A_1.

End of Stage 1. A_1 outputs a challenge message m^*, which is also output by $\widehat{\mathsf{A}_1}$ and given to A_2. $\widehat{\mathsf{A}_1}$ computes $h^* = H(m^*, s^*)$, for a uniformly random independent $s^* \in R_S$. Stage 2 begins.

Stage 2. A_1 continues to make S,KRA and A_2 queries, answered by $\widehat{\mathsf{A}_1}$ as in Stage 1, but can also make designation queries which are answered as follows:

(1) CDV **Queries**. When A_1 makes ith CDV-query pk_i, $\widehat{\mathsf{A}_1}$ searches table T for an entry (sk_j, pk_j, Acc) with $pk_j = pk_i$ (note that this entry is guaranteed to exist in T due to the restriction on A_1 to only query CDV with public keys which have been answered with Acc by a previous KRA query) and answers with $\widehat{\sigma}_i = (h_i, \widehat{r}_{F,i}, \widehat{r}_i, \widehat{s}_i)$, where $h_i = h^*$, $\widehat{r}_i = J(m^*, h^*, \widehat{h}_i) = (\widehat{r}_{i,1}, \dots, \widehat{r}_{i,\alpha})$, $\widehat{h}_i = F_{pk}(\widehat{u}'; \widehat{r}'_{F,i})$ for some fixed $\widehat{u}' = (\widehat{u}'_1, \dots, \widehat{u}'_\alpha) \in (\mathbb{Z}_N^*)^\alpha$ and uniformly random and independent $\widehat{r}'_{F,i} \in R_F$, $\widehat{s}_i = (\widehat{s}_{i,1}, \dots, \widehat{s}_{i,\alpha})$ uniformly random and independent in $(\mathbb{Z}_N^*)^\alpha$, and $\widehat{r}_{F,i} = \mathsf{CSF}((sk_j, pk_j), (\widehat{u}', \widehat{r}'_{F,i}), \widehat{u}_i)$, with $\widehat{u}_i = (\widehat{u}_{i,1}, \dots, \widehat{u}_{i,\alpha})$, where $\widehat{u}_{i,l} = \widehat{s}_{i,l}^e \cdot (h^*)^{-\widehat{r}_{i,l}} \bmod N$.

End of Stage 2. A_2 outputs a decision $d \in \{\mathsf{yes}, \mathsf{no}\}$.

We show that A_1's (and hence also A_2's) view is perfectly simulated in Game no as in Game yes. In particular, the DV signatures $(\widehat{\sigma}_i = (h_i, \widehat{r}_{F,i}, \widehat{r}_i, \widehat{s}_i)$ are distributed identically in both games, for the following reasons. First, note that h^* is computed identically in both games. Second, observe that for each DV signature, $\widehat{r}_i$ and $\widehat{s}_i$ are determined uniquely by $(m^*, h^*, \widehat{u}_i, \widehat{r}_{F,i})$ in the same way in both games, namely

$$\widehat{r}_i = J(h^*, F_{pk}(\widehat{u}_i; \widehat{r}_{F,i}))$$

and

$$\widehat{s}_{i,l} \text{ are the } e\text{th roots of } \widehat{u}_i (h^*)^{\widehat{r}_{i,l}} \bmod N$$

in both games. So it remains to show that (for each DV signature) the pair $(\widehat{u}_i, \widehat{r}_{F,i})$ is identically distributed in both games. In Game yes, $(\widehat{u}_i, \widehat{r}_{F,i})$ is uniform on $G \times R_F$ by definition. In Game no, we have that $(\widehat{u}_i, \widehat{r}_{F,i}) = G(\widehat{s}_i, \widehat{r}'_{F,i})$ for a function $G : (\mathbb{Z}_N^*)^\alpha \times R_F \to (\mathbb{Z}_N^*)^\alpha \times R_F$ defined by:

$$\widehat{u}_{i,l} = \widehat{s}_{i,l}^e \cdot (h^*)^{-J(h^*, F_{pk}(\widehat{u}'; \widehat{r}'_{F,i}))} \bmod N \quad (29)$$
$$\widehat{r}_{F,i} = \mathsf{CSF}((sk, pk), (\widehat{u}', \widehat{r}'_{F,i}), \widehat{u}_i). \quad (30)$$

From the perfectly-trapdoor property of TH we have that mapping $\widehat{r}'_{F,i} \mapsto \mathsf{CSF}((sk, pk), (\widehat{u}', \widehat{r}'_{F,i}), \widehat{u}_i)$ is a permutation on R_F. This and the fact that $\widehat{s}_{i,l} \mapsto \widehat{s}_{i,l}^e \bmod N$ is a permutation of $\mathbb{Z}_N^*$ implies that G is permutation on $(\mathbb{Z}_N^*)^\alpha \times R_F$. So since $(\widehat{s}_i, \widehat{r}'_{F,i})$ is uniform on $(\mathbb{Z}_N^*)^\alpha \times R_F$ by definition, it follows that so is the image pair $(\widehat{u}_i, \widehat{r}_{F,i})$, as required. We conclude that A_2 outputs yes with same probability in both games and hence $C_{\widehat{\mathsf{A}_1}}(\mathsf{A}_1, \mathsf{A}_2) = 0$, as claimed. The run-time of $\widehat{\mathsf{A}_1}$ is the run-time of A_1 plus the time $q_{d1} O(l_J T_N + T_F + T_{CSF} + T_J + T_H + q_{d1} l_{pk} + q_{k1} T_{GKF})$ to answer A_1's CDV and KRA queries. Note that $\widehat{q_{s1}} = q_{s1}$, $\widehat{q_{c1}} = q_{c1}$, and $\widehat{q_{k1}} = 0$, as claimed. $\square$