MTAT.07.006 Research Seminar in Cryptography
Designated Verifier Signature Schemes

Liina Kamm

October 03, 2005

Abstract

This survey gives an overview of the notion of Designated Verifier Signature (DVS) Schemes and the security requirements posed to these schemes.

1 Introduction

The demand for designated verifier signature schemes came from the need to make sure that those and only those verifiers with proper rights could be convinced by the validity of a proof. The solution was first suggested in 1996 in the form of designation of verifiers by Markus Jakobsson, Kazue Sako and Russell Impagliazzo [5]. The notion of undeniable signatures and trap-door commitment schemes are used as a basis for the DVS scheme.

In 2005 Helger Lipmaa, Guilin Wang and Feng Bao showed that the signer can abuse the disavowal protocol of the JSI DVS scheme. They define a new security property - non-delegatability - that is essential for designated verifier signatures. They also propose a new conventional signature scheme, DVS-KW, that is provably unforgeable under a tight reduction to the Decisional Diffie-Hellman problem in the non-programmable random oracle [6]. It is stated in [6] that at this moment the corrected JSI DVS scheme seems to be the most efficient secure disavowable designated verifier signature scheme, while the most efficient secure designated verifier signature scheme seems to be the DVS-KW scheme.

Also in 2005 Rui Zhang, Jun Furukawa, and Hideki Imai proposed the first universal designated verifier signature (UDVS) scheme whose security can be proven without random oracles [12]. This is based on a short signature scheme without random oracles. Also new security definitions are given for UDVS.

2 Preliminaries

In the following survey Alice is the signer, Bob is the designated verifier, Cindy is the third party who wants to gain access to the signed statement, and Dave is a prover. All participants are assumed to be polynomial-time limited and they have to know their own secret keys. In addition, since the protocol is used for practical reasons, the case of Bob being physically or mentally controlled is not viewed, since in that case the attacker has essentially become Bob.

3 Cryptographic Background

In this section short explanations to the cryptographic notions used in the text, are given.

3.1 Bilinear map

Definition 3.1 Let V and W be vector spaces over the same field F. A *linear transformation* is a function $T : V \to W$ such that

1. $T(v + w) = T(v) + T(w)$ for all $v, w \in V$
2. $T(\lambda v) = \lambda T(v)$ for all $v \in V$ and $\lambda \in F$.[8]

Definition 3.2 Let S and U be vector spaces over a field K. A function $B : S \times U \to K$ is called a *bilinear map* if

1. $x \mapsto B(x, y)$ is linear for each $y \in U$

2. $y \mapsto B(x, y)$ is linear for each $x \in S$

That is, B is bilinear if it is linear in each parameter taken separately.[9]

3.2 Decisional Diffie-Hellman Assumption

A *group family* G is a set of finite cyclic groups $G = \{G_p\}$ where p ranges over an infinite index set. $|p|$ denotes the size of binary representation of p. In the following definition it is assumed that there is a polynomial time (in $|p|$) algorithm that given p and two elements in G_p outputs their sum.

An *instance generator*, IG, for G is a randomised algorithm that given an integer n (in unary), runs in polynomial time in n and outputs some random index p and a generator g of G_p. For each n, the instance generator induces a distribution on the set of indices p.

Definition 3.3 Let $G = \{G_p\}$ be a group family. A Decisional Diffie Hellman (DDH) algorithm A for G is a probabilistic polynomial time algorithm satisfying, for some fixed $\alpha > 0$ and sufficiently large n:

$|Pr[A(p, g, g^a, g^b, g^{ab}) = "true"] - Pr[A(p, g, g^a, g^b, g^c) = "true"]| > \frac{1}{n^\alpha}$

where g is a generator of G_p. The probability is over the random choice of $\langle p, g\rangle$ according to the distribution induced by $IG(n)$, the random coice of a, b, c in the range $[1, |G_p|]$ and the random bits used by A. The group family G satisfies the *Decisional Diffie Hellman assumption* if there is no DDH algorithm for G. [1]

3.3 Random oracles

Informally, the random-oracle model for a hash-function h is the model where h is replaced by a uniformly random function. The rational behind the random-oracle model is that by modeling primitives as DES, MD5 or SHA using the strong assumption that they (properly used/modified) behave like random functions, one can build efficient and secure protocols based on these primitives.[7]

When a random oracle is given a query x it does the following:

1. If the oracle has been given the query x before, it responds with the same value it gave the last time.
2. If the oracle hasn't been given the query x before, it generates a random response which has uniform probability of being chosen from anywhere in the oracle's output domain [11] (a uniformly random function $H : \{0,1\}^k \rightarrow \{0,1\}^k$ [7]).

The non-programmable random oracle (NPRO) is known to be strictly weaker than the random oracle (RO) model. In RO, the adversary does not know the secret key, and therefore is forced to program the random oracle to be able to answer successfully to the signature queries. In NPRO, the adversary knows Alice's secret key, and knowing this, can answer successfully to the signature and simulation queries without a need to program the random oracle. A conceptual difference between the two models is that proofs in the RO model work for the "best case" (showing that for every forger there exists a function H_q such that the signature scheme is unforgeable), while proofs in the NPRO model work for the "average case" (showing that the signature scheme is unforgeable for a randomly chosen function $H_q \rightarrow \Omega_{npro}$, independent of the forger).[6]

3.4 Undeniable signatures

Undeniable signature schemes are non-self-authenticating signature schemes, where signatures can only be verified with the signer's consent. However, if a signature is only verifiable with the aid of a signer, a dishonest signer may refuse to authenticate a genuine document. Undeniable signatures solve this problem by adding a new component called the disavowal protocol in addition to the normal components of signature and verification. The probability that a dishonest signer is able to successfully mislead the verifier in either verification or disavowal is 1/p where p is the prime number in the signer's private key.[10]

A suitable group of prime order p and a primitive element g are initially established and made public for use by a set of signers. Consider a particular signer S having a private key x and a corresponding public key g^x. A message $m(\neq 1)$ is signed by Alice to form signature z, which should be equal to m^x.

Bob, receiving z from Alice may wish to establish its validity immediately.

The initial challenge is of the form $z^a(g^x)^b$, where Bob chooses a and b independently and uniformly from the group elements. The response should be formed by Alice, raising the challenge to the multiplicative inverse of x mod p. When Bob computes $m^a g^b$ and finds it equal to the response, then he knows that even if Alice were to have infinite computing power, the probability of z being unequal to m^x (and hence invalid) is at most p^{-1}.

When the value that Bob computes is unequal to the response, the challenge/response protocol should be repeated with independently chosen c and d replacing a and b respectively. Then Bob can use the two responses r_1 and r_2 to test whether $(r_1 g^{-b})^c = (r_2 g^{-d})^a$. Equality means that Alice is answering consistently and z is invalid, with the same high probability as for signature validity; inequality means that Alice is answering improperly.[2]

3.5 Zero Knowledge Proof of Knowledge

An interactive proof system (P, V) for an NP language $L = \{x : \exists w(x, w) \in R\}$ (where R is a polynomial time recognizable relation) is a *proof of knowledge* if there exists a probabilistic polynomial time algorithm K (the *knowledge extractor*) such that for any $x \in L$, and possibly cheating prover P', if $(P', V(x))$ makes V accept (with high probability), then K (given oracle access to P') outputs a w such that $(x, w) \in R$. Formally, we require that there be a negligible function $m(.)$ and a probabilistic polynomial time oracle machine K, such that for every x, w and r, if $P'(x, w, r)$ makes V accept with probability bigger than $m(s)$, then $(x, (Output_K[K(x), P'(x, y, r)])) \in R$ with probability at least $1 - m(s)$.[3]

Definition 3.4 An interactive strategy A is *zero-knowledge* on (inputs from) the set S if, for every feasible (interactive) strategy B^*, there exists a feasible (non-interactive) computation C^* such that the following two probability ensembles are computationally indistinguishable:

1. $\{(A, B^*)(x)\}_{x \in S}$ - the output of B^* after interacting with A on common input $x \in S$;
2. $\{C^*(x)\}_{x \in S}$ - output of C^* on input $x \in S$.[4]

4 Security Notions for Designated Verifier Signature Schemes

In this section the security notions posed to DVS schemes in [6] are brought out.

Secure disavowability - if the DVS scheme has a disavowal protocol then Alice can prove to a third party that a signature is hers and not a simulated signature created by Bob, while she cannot disavow her own signatures.

Non-delegatability - there exists an efficient knowledge extractor that can extract either Alice's or Bob's secret key, when given oracle access to an adversary who can create valid signatures with a high probability. Let $\kappa \in [0, 1]$ be the knowledge error. We say that Δ is (τ, κ)-non-delegatable if there exists a black-box knowledge extractor K that, for every algorithm F and for every valid signature σ, satisfies the following condition: For every $(pk_A, sk_A) \leftarrow Generate$, $(pk_B, sk_B) \leftarrow Generate$ and message m, if F produces a valid signature on m with probability $\varepsilon > \kappa$ then, on input m and on access to the oracle F_m, K produces either sk_A or sk_B in expected time $\tau/(\varepsilon - \kappa)$ (without counting the time to make the oracle queries).

Unforgeability - signatures are verifiable by the designated verifier Bob who rejects it when the signature was not signed by himself or Alice. One can construct conventional signature schemes whose unforgeability is proven by giving a tight reduction to an underlying cryptographic problem; this is achieved by specially avoiding the use of proofs of knowledge.

Non-transferability- given a message-signature pair (m, σ), that is accepted by the designated verifier, and without access to the secret key of the signer, it is computationally infeasible to determine whether the message was signed by the signer, or the signature was simulated by the designated verifier.

Let $\Delta = (Generate, Sign, Simulate, Verify)$ be a designated-verifier signature scheme with the message space M. We say that Δ is perfectly non-transferable if $Sign_{sk_A, pk_B}(m) = Simulate_{sk_B, pk_A}(m)$ as distributions for every

$(pk_A, sk_A) \leftarrow Generate$, $(pk_B, sk_B) \leftarrow Generate$, $H_q \leftarrow \Omega$ ($\Omega = \Omega_{npro}$ or $\Omega = \Omega_{ro}$) and $m \leftarrow M$. Analogously we can define statistically non-transferable and computationally non-transferable schemes.

5 The JSI Designated Verifier Signature Scheme

This section gives a brief overview of the DVS scheme introduced in [5].

Alice wants to prove to Bob that the statement Θ is true. Let Φ_{Bob} be the statement "I know Bob's secret key". Alice will prove to Bob $\Theta \vee \Phi_{Bob}$, who will be convinced that Θ is true (or that his foreign key has been compromised). Since Bob knows his secret key, Cindy will not be convinced that Θ is true, after seeing the proof of $\Theta \vee \Phi_{Bob}$, even if Bob reveals his secret key to her, since Bob is able to produce such a proof himself, independently of whether Θ is true or not.

The notion of undeniable signatures is used as a basis for this DVS scheme. Trap-door commitment schemes (chameleon commitment schemes) are used to construct both interactive and non-interactive designated verifier proofs for undeniable signatures.

Definition 5.1 Let (P_A, P_B) be a protocol for Alice to prove the truth of the statement Θ to Bob. We say that Bob is a *designated verifier* if the following is true: For any protocol (P_A, P'_B, P_C) involving Alice, Bob and Cindy, in which Bob proves the truth of ϑ to Cindy, there is another protocol (P''_B, P_C) such that Bob can perform the calculations of P''_B, and Cindy cannot distinguish transcripts of (P_A, P'_B, P_C) from those of (P''_B, P_C).

Definition 5.2 Let c be a function with input (y_i, w, r), where y_i is the public key of the user who will be able to invert c. The secret key corresponding to y_i is x_i, $w \in W$ is the value committed to and r a random string. We say that c is a *trap-door commitment scheme* if and only if

1. no polynomial-time machine can, given y_i, find a collision (w_1, r_1), (w_2, r_2) such that $c(y_i, w_1, r_1) = c(y_i, w_2, r_2)$
2. no polynomial-time machine can, given y_i and $c(y_i, w, r)$, output w.
3. there is a polynomial-time machine that given any quadruple (x_i, w_1, r_1, w_2) in the set of possible quadruples finds r_2 such that $c(y_i, w_1, r_1) = c(y_i, w_2, r_2)$ for the public key y_i corresponding to the secret key x_i.

5.1 Interactive Designated Verifier Proof of Undeniable Signatures

The designated verifier scheme is based on the generalisation of the confirmation scheme for undeniable signatures (the commitment scheme is not specified). The given scheme can be made designated verifier by letting c be a trap-door commitment scheme, using the public key of the designated verifier. Let p be a large prime, g a generator of G_q, participant i's secret key is x_i and his public key is $y_i = g^{x_i} mod p$. If m is a message, participant i's signature on m will be $s = m^{x_i} mod p$. The used confirmation scheme is the following:

1. Bob uniformly at random selects two numbers a and b from $\mathbb{Z}_q$ and calculates $v = m^a g^b mod p$. Bob sends Alice v.
2. Alice calculates $w = v^{x_A} mod p$. She calculates a commitment c to w and sends c to Bob.
3. Bob sends (m, s, a, b) to Alice, who verifies that v is of the right form.
4. Alice decommits to c by sending w and any possible random string r used for the commitment to Bob. Bob verifies that $w = s^a y_A^b mod p$ and that the commitment c was correctly formed.

5.2 Non-interactive Designated Verifier Proofs

This scheme is a bridge between publicly verifiable digital signatures and undeniable signatures - it limits who can verify the signature without help from the prover and without interaction. In contrast to publicly verifiable signatures that are made designated verifier, they can be used for contracts, etc., as their validity can be verified when the prover agrees to this. A general method to transform ordinary three-move zero-knowledge protocols

to non-interactive designated verifier proofs is given below. The basis of the method is the Fiat-Shamir technique for making an ordinary three-move zero-knowledge protocol non-interactive, while preserving the security of the protocol in a practical manner. The same denotation as in the previous section is used to give a corresponding non-interactive designated verifier proof:

Constructing a proof. The prover, Alice, selects w, r, $t \in_u \mathbb{Z}_q$ and calculates $c = g^w y_B^r modp$, $G = g^t modp$, $M = m^t modp$, $h = hash_q(c, G, M)$, $d = t + x_A(h + w) modq$, where $hash_q$ gives a hashed value in $\mathbb{Z}_q$. The prover sends (w, r, G, M, d) to the verifier, Bob.

Verifying a proof. The designated verifier calculates $c = g^w y_B^r modp$, $h = hash_q(c, G, M)$. And verifies that $G_{y_A}^{h+w} = g^d modp$ and $M_s^{h+w} = m^d modp$.

Simulating transcripts. The designated verifier can simulate correct transcripts by selecting $d, \alpha, \beta \in_u \mathbb{Z}_q$ and calculate $c = g^\alpha modp$, $G = g^d y_A^{-\beta} modp$, $M = m^d s^{-\beta} modp$, $h = hash_q(c, G, M)$, $w = \beta - h \bmod q$, $r = (\alpha - w) x_B^{-1} modq$.

5.3 Extension to Multiple Designated Verifiers

If Alice wants to convince a set of n verifiers, $\{Bob_i\}_{i=1}^n$ but only these, the trivial approach would be for Alice to convince each individual verifier Bob_i in an individual proof. The following solution is proposed as an alternative to the trivial case. The previously described protocol is used with the modification that c is a function that is one-way to each coalition of less than n of the designated verifiers, but invertible if they all cooperate. This can be done by letting the secret key be distributed among all the n designated verifiers so that they all need to cooperate to calculate it. It is not necessary for the designated verifiers to share a secret in advance. Each designated verifier would be convinced by the proof as long as he knows that his share of the secret key has not been compromised. No outsider Cindy would be able to receive conviction because the set of verifiers $\{Bob\}_{i=1}^n$ could have cooperated to cheat her. They could do this without revealing their personal shares of the secret key to each other.

5.4 The Notion of Strong Designated Verifier

Definition 5.3 Let (P_A, P_B) be a protocol for Alice to prove the truth of the statement Θ to Bob. We say that Bob is a *strong designated verifier* if the following is true: For any protocol (P_A, P_B, P_D, P_C) involving Alice, Bob, Dave and Cindy in which Dave proves the truth of ϑ to Cindy, there is another protocol (P'_D, P_C) such that Dave can perform calculations of P'_D and Cindy cannot distinguish transcripts of (P_A, P_B, P_D, P_C) from those of (P'_D, P_C).

This notion is necessary for an honest Bob. It is generally assumed that Cindy will not believe $\Theta \vee \Phi_{Bob}$, as she knows that Bob could have produced such a transcript himself. If Bob is honest then Cindy would be convinced that Θ is true upon seeing a proof of $\Theta \vee \Phi_{Bob}$. In order to make protocols strong designated verifier, transcripts can be probabilistically encrypted using the public key of the intended verifier. No "honest" participant will agree to decrypt this type of ciphertext. Since Dave will not be able to present the decrypted transcripts to Cindy, and she cannot distinguish encrypted transcripts from random strings of the same length and distribution, Dave will be able to produce transcripts (P_B, P'_D, P_C) that Cindy cannot distinguish from transcripts of (P_A, P_B, P_D, P_C).

5.5 Attack on the JSI DVS Scheme and the Corrected JSI Scheme

The proposed DVS scheme is open to a disavowability attack according to [6]. A malicious Alice can generate signatures exactly from the same distribution as Bob. So the scheme is perfectly non-transferable and thus also not disavowable.

Alice computes a signature $(\overline{s}; w, t, G, \overline{M}, z)$ for a message m, with $\overline{s} \neq m^{x_A}$, as follows. She uniformly elects four random numbers $w, t, r, \overline{r} \in \mathbb{Z}_q$ and then sets $c = g^w y_B^t modp$, $G = g^r modp$, $\overline{M} = m^{\overline{r}} modp$, $h = H_q(c, G, \overline{M})$, $z = r + (h + w) x_A modq$ and $\overline{s} = m^{x_A} \cdot m^{(r-\overline{r})/(h+w) modq} modp$. Alice then sends a message-signature pair $(m, \overline{s})$ with $\overline{\sigma} = (\overline{s}, P = (w, t, G, \overline{M}, z))$ to Bob. Bob will believe that $\overline{s}$ is Alice's signature for message m. In later disputes, Alice can convince a third party (e.g., a judge) that $\overline{s}$ was simulated by Bob, by us-

ing a standard disavowal protocol to show that $log_g y_A \neq log_m \overline{s}$.

It can be shown that this scheme is unforgeable, non-delegatable, computationally non-transferable and securely disavowable after a trivial fix of adding some additional variables under the used hash value, by following the usual proof of knowledge methodology.

The first countermeasure is to force Alice to provide an additional proof of knowledge that $log_m M = log_g G$. This, however, increases the signature length. The second possibility is to include s (together with pk_A and pk_B) to the input of the hash function. This turns out to be sufficient though it makes the scheme computationally but not perfectly non-transferable.

6 The DVS Scheme with Tight Reduction to the Decisional Diffie-Hellman Problem in the Non-programmable Random Oracle

A new DVS Scheme DVS-KW is proposed in [6], where the signer presents a designated verifier proof that his public key is a Decisional Diffie-Hellman (DDH) tuple. The unforgeability of this scheme is proved by providing a tight reduction to the underlying cryptographic problem (DDH) in the non-programmable random oracle (NPRO) model. This scheme is also non-delegatable (though this proof is in the programmable random oracle model and has a larger security degradation due to the involved proof-of-knowledge property), correct and perfectly non-transferable, unforgeable in the non-programmable random oracle model.

Let p, q be two large primes, such that $q|(p-1)$ and G_q is a multiplicative subgroup of $\mathbb{Z}_p^*$. Let $g_1, g_2 \in G_q$ be two elements such that nobody knows the mutual discrete logarithms of g_1 and g_2. In the DVS-KW DVS scheme Alice proves to Bob that $(g_1, g_2, y_{1A}, y_{2A})$ is a Decisional Diffie-Hellman tuple, where $x_i \leftarrow_r \mathbb{Z}_q$ is i's private key and $pk_i = (g_1, g_2, y_{1i}, y_{2i})$ is i's public key with $y_{1i} = g_1^{x_i}$ and $y_{2i} = g_2^{x_i}$. This proof is made designated-verifier by using the same trick as in the JSI scheme and non-interactive by using a non-programmable random oracle H_q with outputs from $\mathbb{Z}_q$. In particular, the random oracle H_q can be chosen at the same stage as other system parameters, g_1 and g_2. The description of the full DVS-KW scheme follows:

$Sign_{sk_A,pk_B}(m)$: Alice generates random $r, w, t \leftarrow \mathbb{Z}_q$, and sets $a_1 = g_1^r mod p$, $a_2 = g_2^r mod p$,$c = g_1^w y_{1B}^t mod p$, $h = H_q(pk_A, pk_B, a_1, a_2, c, m)$ and $z = r + (h + w)x_A mod q$. She outputs the signature $\sigma = (w, t, h, z)$.

$Simulate_{sk_B,pk_A}(m)$: By selecting three random numbers $z, \alpha, \beta \leftarrow_r \mathbb{Z}_q$, Bob creates $\sigma = (w, t, h, z)$ for any message m as follows: $(a_1, a_2) = (g_1^z y_{1A}^{-\beta} mod p, g_2^z y_{2A}^{-\beta} mod p)$, $h = H_q(pk_A, pk_B, a_1, a_2, g_1^\alpha mod p, m)$, $w = \beta - h \bmod q$, $t = (\alpha - w)x_B^{-1} mod q$.

$Verify_{pk_A,pk_B}(m; w, t, h, z)$: The verifier checks whether $h = H_q(pk_A, pk_B, g_1^z y_{1A}^{-(h+w)} \bmod p, g_2^z y_{2A}^{-(h+w)} \bmod p, g_1^w y_{1B}^t mod p, m)$.

DVS-KW can be seen as a proof of concept, showing how to design DVS schemes that have a tight reduction in the unforgeability proof and are still non-delegatable. DVS-KW is more efficient than the JSI scheme, and it does not allow the signer to disavow simulated signatures.

7 Universal Designated Verifier Signature without Random Oracles

A UDVS scheme without random oracles is proposed in [12]. It is based on a short signature scheme without random oracles. Also new security notions have been given for UDVS.

7.1 Short Signature Scheme without Random Oracles

This signature scheme is used as an important building block of UDVS.

Let (G_1, G_2) be bilinear groups where $|G_1| = |G_2| = p$ for some large prime p m is the message to be signed and is encoded as an element of $\mathbb{Z}_p^*$.

Generate: Pick a random generator $g_2 \in G_2$ and set $g_1 = \psi(g_2)$. Pick $x, y \leftarrow \mathbb{Z}_p^*$, and com-

pute $u = g_2^x$ and $v = g_2^y$. For fast verification, also compute $z = e(g_1, g_2) \in G_T$. The public key is (g_1, g_2, u, v, z) and the secret key is (x, y).

Sign: Given a secret key $(x, y) \in (\mathbb{Z}_p^*)^2$ and a message $m \in \mathbb{Z}_p^*$, pick $r = \mathbb{Z}_p^*$. If $x+r+ym = 0 \bmod p$, try again with a different random r. Compute $\sigma = g_1^{1/(x+r+ym)} \in G_1$. The signature is (σ, r).

Verify: Give the public key (g_1, g_2, u, v, z), a message $m \in \mathbb{Z}_p^*$, and a signature (σ, r), accept if $e(\sigma, u \cdot g_2^r \cdot v^m) = z$, otherwise, reject.

7.2 Model of UDVS

A universal designated verifier signature (UDVS) scheme UDVS = (CPG, SKG,VKG, S, PV,DS,DV, P_{KR}).

1. Common Parameter Generation CPG - a probabilistic algorithm, given a security parameter k, outputs a string cp consisting of common scheme parameters (publicly shared by all users).

2. Signer Key Generation SKG - a probabilistic algorithm, on input a common parameter string cp, outputs a secret/public key-pair (sk_a, pk_a) for Alice.

3. Verifier Key Generation VKG - a probabilistic algorithm, on input a common parameter string cp, outputs a secret/public key-pair (sk_b, pk_b) for Bob.

4. Signing S - possibly a probabilistic algorithm, on input Alice's secret key sk_a and a message m, outputs Alice's public verifiable (PV) signature s.

5. Public Verification PV - a deterministic algorithm, on input Alice's public key pk_a and message/PV-signature pair (m, s), outputs verification result $d \in (acc, rej)$.

6. Designation DS - possibly a probabilistic algorithm, on input Alice's public key pk_a, Bob's public key pk_b, and a message/PV-signature pair (m, s), outputs Designated-Verifier (DV) signature $\overline{s}$.

7. Designated Verification DV - a deterministic algorithm, on input Alice's public key pk_a, Bob's secret key sk_b, and messge/DV-signature pair $(m, \overline{s})$, outputs verification decision acc or rej.

8. Verifier Key-Registration P_{KR}(KR,V) - a protocol between a Key Registration Authority KR and a Verifier V. The verifier registers a verifier's public key. On common input cp, KR and V interact with messages sent each other. At the end of the protocol, KR outputs a pair $(pk_b, Auth)$, where pk_b is the public key of V and Auth $\in$ {acc, rej} indicates whether or not the key-registration is successful.

7.3 Security Notions for UDVS

Strong DV-Unforgeability - There are two types of unforgeability to consider: Public Verifiable signature unforgeability (PV-unforgeability), the security for the signer, which states that anyone should not be able to forge a PV-signature of the signer. Designated Verifier signature unforgeability (DV-unforgeability), the security for the designated verifier, which states that for any message, an adversary without a PV-signature should be unable to convince a designated verifier of holding such a PV-signature. DV-unforgeability always implies PV-unforgeability, because anyone able to forge a PV-signature can transform it into a DV-signature. Thus it is enough to consider only DV-unforgeability.

Non-Transferability - A is an attacker that tries to brag about its interaction with the signature holder. S is a simulator that simulates the output of A. S is able to access A as a black-box. D is a distinguisher that tries to distinguish whether a given output is of A or of S. We say a UDVS scheme is *unconditionally non-transferable against adaptive chosen public key attack and chosen message attack* (NT-CPKMA), if there exists S such that for every A, every computationally unbounded D distinguishes outputs of A and S on any challenge message m^* with only probability $negl(k)$, where the probability is taken over the coin toss of key generation algorithms, S, A, S and D. A is able to access to Designation oracle with respect to any message (including the challenge message) before the challenge message is determined. This helps the adversary adaptively choose the challenge message.

7.4 Model of UDVS without Random Oracles

For simplicity, the verifier key registration protocol is omitted, since in practice this is needed to run only once.

1. Common Parameter Generation CPG: Choose a bilinear group pair which is denoted by a description string $Str_D : (G_1, G_2)$ of prime order $|G_1| = |G_2| = p$ with a bilinear map $e : G_1 \times G_2 \rightarrow G_T$ and an isomorphism $\psi : G_2 \rightarrow G_1$. Choose a random generator $g_2 \in G_2$ and compute $g_1 = \psi(g_2) \in G_1$. Then the common parameter is $cp = (Str_D, g_1, g_2)$.

2. Signer Key Generation SKG: Given cp, pick random $x_1, y_1 \leftarrow \mathbb{Z}_p^*$, compute $u_1 = g_2^{x_1}$ and $v_1 = g_2^{y_1}$. Specially, for speeding up the verification, one may also compute $z \leftarrow e(g_1, g_2) \in G_T$. The public key is $pk_a = (cp, u_1, v_1, z)$, the secret key is $sk_a = (x_1, y_1)$.

3. Verifier Key Generation VKG: Given cp, pick random $x_3, y_3 \leftarrow \mathbb{Z}_p^*$. Compute $u_3 = g_2^{x_3}$ and $v_3 = g_2^{y_3}$

4. The public key is $pk_b = (cp, u_3, v_3)$ and the secret key is $sk_b = (x_3, y_3)$.

5. Signing S: Given the signer's secret key (cp, x_1, y_1) and a message m, select $r \leftarrow \mathbb{Z}_p^*$. If $x_1 + r + my_1 = 0 \bmod p$, restart. Compute $\sigma = g_1^{1/(x_1+r+my_1)}$ and output $s = (\sigma, r)$ as the PV-signature.

6. Public Verification PV: Given the signer's public key (cp, u_1, v_1, z), and a message/PV-signature pair (m, s), accept only if $e(\sigma, u_1 \cdot g_2^r \cdot v_1^m) = z$; otherwise reject.

7. Designation DS: Given the signer's public key (cp, u_1, v_1), a verifier's public key (cp, u_3, v_3) and a message/PV-signature pair (m, s), where $s = (\sigma, r)$, let $h = g_2^r$ and compute $d = e(\psi(u_3), v_3^r) \in G_T$. Then the DV-signature is $\overline{s} = (\sigma, h, d)$.

8. Designated Verification DV: Given a signer's public key (cp, u_1, v_1), a verifier's secret key (x_3, y_3), and message/DV-signature pair $(m, \overline{s})$, accept only if the following two equations hold simultaneously: $z = e(\sigma, u_1 \cdot h \cdot v_1^m)$ and $d = e(\psi(u_3), h^{y_3})$. Otherwise, reject.

The scheme is correct, unforgeable against adaptive chosen public key attack and chosen message attack for designated verifier, and unconditionally non-transferable. It is, however, delegatable.

References

[1] D. Boneh. The decision Diffie-Hellman problem, In Proceedings of the Third Algorithmic Number Theory Symposium, Lecture Notes in Computer Science, Vol. 1423, Springer-Verlag, pp. 48–63, 1998. http://theory.stanford.edu/ dabo/ papers/DDH.ps.gz

[2] David Chaum, Hans van Antwerpen. Undeniable signatures, Advances in Cryptology - Crypto '89, Springer-Verlag (1990), 212-216. http://dsns.csie.nctu.edu.tw/ research/crypto/HTML/PDF/C89/212.PDF

[3] CSE208: Advanced cryptography, Lecture 6: Zero Knowledge Proofs of Knowledge, Fall 2002 http://www.cse.ucsd.edu/classes/ fa02/cse208/lec6.html

[4] Oded Goldreich. Zero-knowledge twenty years after its invention. Technical Report 2002. http://citeseer.ist.psu.edu/ goldreich02zeroknowledge.html

[5] Markus Jakobsson, Kazue Sako and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. Proc. Eurocrypt'96, p. 143–154, Springer-Verlag, 1996.

[6] Helger Lipmaa, Guilin Wang and Feng Bao. Designated Verifier Signature Schemes: Attacks, New Security Notions and A New Construction. In Luis Caires, Guiseppe F. Italiano, Luis Monteiro, Catuscia Palamidessi and Moti Yung, editors, The 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005, volume

3580 of Lecture Notes in Computer Science, p. 459–471, Lisboa, Portugal, July 11–15, 2005. Springer-Verlag.

[7] Jesper Buus Nielsen. On Protocol Security in the Cryptographic Model, BRICS Dissertation Series DS-03-8 ISSN 1396-7002, Department of Computer Science, University of Aarhus, August 2003 http://citeseer.ist.psu.edu/nielsen03protocol.html

[8] PlanetMath http://planetmath.org/encyclopedia/LinearTransformation.html

[9] PlanetMath http://planetmath.org/encyclopedia/BilinearForm.html

[10] RSA Laboratories, Home: Crypto FAQ: Chapter 7 Miscellaneous Topics, 7.8 What is an undeniable signature scheme? http://www.rsasecurity.com/ rsalabs/node.asp?id=2344

[11] Random oracle From Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Random_oracle

[12] Rui Zhang, Jun Furukawa, and Hideki Imai. Short Signature and Universal Designated Verifier Signature Without Random Oracles. Applied Cryptography and Network Security, Third International Conference, ANCS 2005, volume 3531 of Lecture Notes in Computer Science, p. 483-498, New York, June 7-10, 2005. Springer-Verlag.