MTAT.07.007 Graduate Seminar in Cryptography
Universal Designated Verifier Signature without Random Oracles

Liina Kamm

March 27, 2006

Abstract

This survey gives an overview of a Universal Designated Verifier Signature (UDVS) without random oracles. It also discusses the security requirements set to the scheme.

1 Introduction

The demand for designated verifier signature schemes came from the need to make sure that only verifiers with proper rights could be convinced by the validity of a proof and they should not be able to present the signatures to other parties (e.g. certificates for hospital records, income summaries). The solution was first suggested in 1996 in the form of designation of verifiers [1].

Universal designated verifier signature (UDVS) is an important tool to protect the privacy of the signature holder from dissemination of signatures by verifiers. UDVS schemes are signature schemes with additional functionality where any holder of the signature alone can transform the signature to a non-interactive proof statement for a desired designated verifier using the knowledge of the signature, such that the designated verifier can verify the message is signed by the signer but cannot prove the same fact to a third party, since he can also produce such a proof statement using his secret key [5].

In this case 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 secret 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.[1]

The good properties of UDVS make it an important tool to prevent dissemination of digital signatures in user certification systems. It is thus desirable to have a rigorous model and corresponding formal analysis for UDVS schemes. However, most of the UDVS schemes are provably secure in the random oracle model. Random oracle model is a formal model in analyzing cryptographic schemes, where a hash function is considered as a black-box that contains a random function. However, many impossibility results have shown that security in the random oracle model does not imply the security in the real world in that a scheme can be secure in the random oracle model and yet be broken without violating any particular intractability assumption, and without breaking the underlying hash functions.[5]

In 2005 Rui Zhang, Jun Furukawa, and Hideki Imai proposed the first UDVS scheme [5] whose security can be proven without random oracles. This is based on a short signature scheme without random oracles. Also new security definitions are given for UDVS, which allow the adversary to behave more adaptively in oracle accessing.

In 2005 Helger Lipmaa, Guilin Wang and Feng Bao defined a new security property - non-delegatability - that is essential for designated verifier signatures [3]. Together with the results of [2], they show that most of the proposed DVS schemes are delegatable. However the UDVS scheme proposed by Zhang-Furukawa-Imai may still be secure in practice, since the only party who can delegate signing is the designated verifier, who may not have motivation to do so. [3]

2 Cryptographic Background

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

2.1 Signature Schemes

A signature scheme consists of three algorithms: $SIG = (KG, Sig, Ver)$. $KG(1^k)$ is the key generation algorithm that outputs a pair of keys (pk, sk). The signing algorithm S takes sk and a message m from the associated message space $\mathcal{M}$, outputs a signature s. Ver is the deterministic verification algorithm, that takes pk, a message m, and s as input, outputs acc for accepted or rej for rejected as the verification result.

In this scheme the security definition called strong existential unforgeability against adaptive chosen message attack (sEUF-CMA) is considered. Let's view the following game:

Setup: KG is run, generates a key pair (pk, sk). pk is given to the adversary $\mathcal{A}$ and sk is given to a challenger.

Training: The adversary $\mathcal{A}$ requests signatures on at most q_s messages $m_1, m_2, ..., m_{q_s} \in \mathcal{M}$ chosen adaptively by itself. The challenger responds with the corresponding signature $s_i = Sig(m_i)$, $i = 1, ..., q_s$.

Forge: Eventually, the adversary $\mathcal{A}$ outputs a pair (m, s) and wins the game if $(m, s) \notin \{(m_1, s_1), ..., (m_{q_s}, s_{q_s})\}$ and $Ver(m, s) = acc$.

Let Adv be the probability that the adversary wins the above game, which is taken over the coin toss made by $\mathcal{A}$ and the Challenger.

Definition 2.1 An adversary (q_s, t, ε)-breaks the signature scheme if $\mathcal{A}$ makes at most q_s signature queries, runs in time at most t and Adv is at least ε. A signature scheme is (q_s, t, ε)-strong existentially unforgeable under an adaptive chosen message attack if no PPT adversary (q_s, t, ε)-breaks it.

2.2 Bilinear Groups

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p, where Computational Diffie-Hellman problem (CDH) is considered hard. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$ with the following properties:

1. Bilinear: for all $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.

2. Non-degenerate: $e(g_1, g_2) \neq 1$.

Two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p are a bilinear map group pair if there is an additional group $\mathbb{G}_T$ with $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$, such that there exist a bilinear map e and an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$.

2.3 Strong Diffie-Hellman Assumption

Definition 2.2 Let g_1 and g_2 be as above with $g_1 = \psi(g_2)$. A PPT adversary $\mathcal{A}$ (q, t, ε)-breaks the Strong Diffie-Hellman problem in $(\mathbb{G}_1, \mathbb{G}_2)$ if after given q-tuples of $g_2^{(x^i)}$ with $1 \leq i \leq q$ and running time t, has the probability ε of outputting a pair $(c, g_1^{\frac{1}{(x+c)}})$ where $c \in \mathbb{Z}_p^*$. The probability is taken over the random choice of generator g_2, the choice of $x \in \mathbb{Z}_p^*$, and internal random coins of $\mathcal{A}$. The (q, t, ε)-Strong Diffie-Hellman assumption holds if no PPT algorithm solves the SDH problem.

3 Short Signature Scheme

This short signature scheme without random oracles is used as an important building block of UDVS.

Let $(\mathbb{G}_1, \mathbb{G}_2)$ be bilinear groups where $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ 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 $g_2 \in \mathbb{G}_2$ and set $g_1 = \psi(g_2)$. Pick $x, y \leftarrow \mathbb{Z}_p^*$, and compute $u = g_2^x$ and $v = g_2^y$. For fast verification, also compute $z = e(g_1, g_2) \in \mathbb{G}_T$. 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 $\sigma = g_1^{1/(x+r+ym)} \in \mathbb{G}_1$. 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.

If a collision-free hash function is applied to m, whose output can be encoded as an element of

$\mathbb{Z}_p^*$, one can still prove the security against (sEUF-CMA) of the resulting signature scheme without random oracles.

4 General Model of UDVS

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

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 key-pair (sk_s, pk_s) for $Signer$.

3. **Verifier Key Generation** VKG - a probabilistic algorithm, on input a common parameter string cp, outputs a key-pair (sk_v, pk_v) for $Verifier$.

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

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

6. **Designation** DS - possibly a probabilistic algorithm, on input Signer's public key pk_s, Verifier's public key pk_v, and a message/PV-signature pair (m, s), outputs Designated-Verifier (DV) signature $\hat{s}$.

7. **Designated Verification** DV - a deterministic algorithm, on input Signer's public key pk_s, Bob's secret key sk_v, and messge/DV-signature pair $(m, \hat{s})$, 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 $(pk_v, Auth)$, where pk_v is the public key of V and $Auth \in \{acc, rej\}$ indicates whether or not the key-registration is successful.

The key registration is necessary in practice because the signature receiver can easily convince a third party the validity of a signature by claiming he is the owner of the third party's public key and present the UDVS to the third party.

5 Security Notions for UDVS

5.1 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 noone should be able to forge a PV-signature of the signer. Designated Verifier signature unforgeability (DV-unforgeability) is 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.

In this scheme the security definition called strong existential unforgeability for designated verifier against adaptive chosen public key and chosen message attack (sEUF-DV-CPKMA) is considered. It is defined via the following game:

$\mathcal{F}$, a forger attacking the DV-unforgeability of UDVS, plays the following game with a challenger.

Setup: The key generation algorithms are run, $cp \leftarrow CPG(1^k)$, $(pk_s, sk_s) \leftarrow SKG(cp)$, $(pk_{v_i}, sk_{v_i}) \leftarrow PKG_i(cp)$ for $1 \leq i \leq n$. All public keys are given to $\mathcal{F}$ and the challenger. All secret keys are given to the challenger. The challenger maintains two lists: M and $\hat{S}$, initially empty. The challenger additionally maintains a list L which consists of all the public keys $\{pk_{v_i}\}$ for $1 \leq i \leq n$, which are assumed to be already registered.

Training: $\mathcal{F}$ may adaptively issue q_s times signing queries, q_d times designation queries, q_v times Designated Verification queries, and up to $n-1$ of key registration queries to the challenger. However, once pk_{v_i} is queried by $\mathcal{F}$, the challenger neglects further designated verification queries with respect to verifier V_i's public key pk_{v_i}.

1. On a Signing query by $\mathcal{F}$ on m, the challenger returns the corresponding signature $s = S(sk_s, m)$ to $\mathcal{F}$ and adds m to M.

2. On Designation query by $\mathcal{F}$ on m and pk_v, the challenger first computes the corresponding PV-signature $s = S(sk_s, m)$, and then $\hat{s} = DS(pk_s, pk_v, m, s)$. The challenger then adds $(m, \hat{s})$ to $\hat{S}$, and returns $\hat{s}$ to $\mathcal{F}$.

3. On a Designated Verification oracle query by $\mathcal{F}$ on $(m, \hat{s})$ on pk_{v_i}, the challenger runs the designated verification algorithm $DV(pk_s, sk_{v_i}, m, \hat{s})$ and returns the corresponding verification result to $\mathcal{F}$.

4. On a Key Registration query by $\mathcal{F}$ on pk_v, the challenger sends the corresponding secret key sk_v to $\mathcal{F}$, and deletes pk_v from L.

Forge: Denote $\mathcal{F}$'s running time as t. $\mathcal{F}$ outputs $(m^*, \hat{s}^*)$ and wins the game if $DV(pk_s, sk_{v^*}, m^*, \hat{s}^*) = acc$ with $(m^*, \hat{s}^*) \notin \hat{S}$, where $pk_{v^*} \in L$, and $m^* \in M$ is a valid message from the message space.

Suppose q_s, q_d, q_v and t are all polynomially bounded. A UDVS signature is secure against sEUF-DV-CPKMA if no any probabilistic polynomial time $\mathcal{F}$, can win the above game with non-negligible probability.

In this notion the adversaries are allowed to adaptively corrupt designated verifiers and adaptively choose the target designated verifier, which reflects more essence of real world adversaries.

The adversary is allowed to make designation queries which captures the attack scenario where a real world adversary may obtain $(m, \hat{s})$ without knowing the corresponding PV-signature s.

5.2 Non-Transferability

$\mathcal{A}$ is an attacker that tries to brag about its interaction with the signature holder. $\mathcal{S}$ is a simulator that simulates the output of $\mathcal{A}$. $\mathcal{S}$ is able to access $\mathcal{A}$ as a black-box. $\mathcal{D}$ is a distinguisher that tries to distinguish whether a given output is of $\mathcal{A}$ or of $\mathcal{S}$.

Setup: $cp \leftarrow CPG(1^k)$. $(pk_s, sk_s) \leftarrow SKG(cp)$. KR is a Key Registration oracle, who maintains a list of verifier's public keys, initially empty.

Training: $\mathcal{A}$ and $\mathcal{S}$ are allowed to have the following resources:

1. $\mathcal{A}$ and $\mathcal{S}$ are allowed to access the Signing oracle $S(sk_s, \cdot)$ up to q_s times and q'_s times, respectively. However, after the challenge message m^* is output, they may not access to Signing oracle S with respect to this challenge message.

2. $\mathcal{S}$ and $\mathcal{A}$ can output the challenge message m^* at an arbitrary time but only once.

3. $\mathcal{A}$ and $\mathcal{S}$ are allowed to access KR up to q_k and q'_k times, respectively.

4. $\mathcal{A}$ and $\mathcal{S}$ are allowed to access $\mathcal{D}$ up to q_c and q'_c times, respectively.

5. $\mathcal{A}$ is allowed to access to designation oracles $DS(pk_{v^i}, \cdot)$ up to q_d times as long as $(pk_{v^i}$ is correctly registered.

6. $\mathcal{S}$ is NOT allowed to access DS.

Guess: The running time of $\mathcal{A}$, $\mathcal{S}$ are t and t', respectively. Finally, $\mathcal{A}$ and $\mathcal{S}$ return their outputs to $\mathcal{D}$ with respect to m^*. $\mathcal{D}$ decides whether this output is of $\mathcal{A}$ or of $\mathcal{S}$.

We say a UDVS scheme is *unconditionally non-transferable against adaptive chosen public key attack and chosen message attack* (NT-CPKMA), if there exists $\mathcal{S}$ such that for every $\mathcal{A}$, every computationally unbounded $\mathcal{D}$ distinguishes outputs of $\mathcal{A}$ and $\mathcal{S}$ on any challenge message m^* with only probability $negl(k)$, where the probability is taken over the coin toss of key generation algorithms, S, $\mathcal{A}$, $\mathcal{S}$ and $\mathcal{D}$. $\mathcal{A}$ 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.

Basically, it means that 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 [2].

6 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 $Str_D : (\mathbb{G}_1, \mathbb{G}_2)$ of prime order $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Choose a random generator $g_2 \in \mathbb{G}_2$ and compute $g_1 = \psi(g_2) \in \mathbb{G}_1$. 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 $z \leftarrow e(g_1, g_2) \in \mathbb{G}_T$. The public key is $pk_s = (cp, u_1, v_1, z)$, the secret key is $sk_s = (x_1, y_1)$.

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 $v_3 = g_2^{y_3}$ The public key is $pk_v = (cp, u_3, v_3)$ and the secret key is $sk_v = (x_3, y_3)$.

4. **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.

5. **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.

6. **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 $d = e(\psi(u_3), v_3^r) \in \mathbb{G}_T$. Then the DV-signature is $\hat{s} = (\sigma, h, d)$.

7. **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 $(m, \hat{s})$, 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, strong existentially unforgeable against adaptive chosen public key and chosen message attack for designated verifier, and unconditionally non-transferable. The security proofs are given in [5].

7 On Delegatability

In [3] a new security property for DVS schemes - non-delegatability - is defined. This means that either the signer or one of the designated verifiers can delegate the signing rights to a third party T without disclosing his or her secret key.

Non-delegatability - there exists an efficient knowledge extractor that can extract either the Signer's or the Verifier'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 $\mathcal{K}$ that, for every algorithm $\mathcal{F}$ and for every valid signature σ, satisfies the following condition: For every $(pk_s, sk_s) \leftarrow Generate$, $(pk_v, sk_v) \leftarrow Generate$ and message m, if $\mathcal{F}$ produces a valid signature on m with probability $\varepsilon > \kappa$ then, on input m and on access to the oracle $\mathcal{F}_m$, $\mathcal{K}$ produces either sk_s or sk_v in expected time $\frac{\tau}{\varepsilon - \kappa}$ (without counting the time to make the oracle queries).

In [2] the delegatability notion is specified further with verifier-only delegatability. This means that one of the designated verifiers (or even only the coalition of all verifiers) can delegate the signing right to a third party without disclosing his or her secret key, while the signer cannot do it.

Verifier-only delegatability: $\mathcal{F}_m$ denotes $\mathcal{F}$ with m as its input. Oracle calls are counted as one step. More precisely, let $\kappa \in [0, 1]$ be the knowledge error. We say that δ is verifier-only (τ, κ)-delegatable if it is not (τ, κ)-non-delegatable and there exists a black-box knowledge extractor $\mathcal{K}$ that, for every algorithm $\mathcal{F}$ and for every message $m \in \mathcal{M}$ satisfies the following condition: for every $(sk_s, pk_s) \leftarrow KeyGen, (sk_{v_1}, pk_{v_1}) \leftarrow KeyGen, ..., (sk_{v_n}, pk_{v_n}) \leftarrow KeyGen$, for every bitstring d (delegation token) that does not depend on sk_{v_i} for any $i \in [1, n]$, and for any message m, if $\mathcal{F}$ produces a valid signature on m with probability $\varepsilon > \kappa$ then, on input m and on access to the oracle

$\mathcal{F}_m$, $\mathcal{K}$ produces sk_{v_i} for some $i \in [1, n]$ in expected time $\frac{\tau}{\varepsilon - \kappa}$ (without counting the time to make the oracle queries). $\mathcal{F}$'s probability is taken over the choice of her random coins.

In [2] an attack on the Zhang-Furukawa-Imai UDVS scheme is brought out.

$\mathbf{Simul}_{pk_s, sk_v}(m)$: Pick a random $s \in \mathbb{Z}_q^*$ and compute $\sigma' \leftarrow g_2^s$, $h \leftarrow g_2^{1/s} u_1^{-1} v_1^{-m}$ and $d \leftarrow \langle g_1, h \rangle^{x_3 y_3}$. Return $\sigma \leftarrow (\sigma', h, d)$.

In the simulation algorithm, the designated verifier can compute d as $d \leftarrow \langle g_1^{x_3 y_3}, h \rangle$. Thus, the designated verifier can reveal $g_1^{x_3 y_3}$, and therefore this scheme is delegatable by the verifier. The delegation token does not depend on the signer.

Thus the Zhang-Furukawa-Imai UDVS scheme is verifier-only delegatable.

References

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

[2] Yong Li, Helger Lipmaa and Dingyi Pei. On Delegatability of Four Designated Verifier Signatures. Information and Communications Security: 7th International Conference, ICICS 2005, volume 3783 of Lecture Notes in Computer Science, p. 61–71, Beijing, China, December 10–13, 2005. Springer-Verlag.

[3] 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.

[4] 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

[5] 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.