On the Soundness of Restricted Universal Designated Verifier Signatures and Dedicated Signatures

How to prove the possession of an Elgamal/DSA signature

Fabien Laguillaumie[1] and Damien Vergnaud[⋆2]

[1] GREYC – Université de Caen
Boulevard du Maréchal Juin - BP 5186 - 14032 Caen Cedex, France
fabien.laguillaumie@info.unicaen.fr
[2] École normale supérieure
Département d'informatique, 45 rue d'Ulm, 75230 Paris cedex 05, France
damien.vergnaud@di.ens.fr

Abstract. In 2006, Huang, Susilo, Mu and Zhang proposed the concept of *restricted universal designated verifier signatures* while Klonowski, Kubiak, Kutylowski and Lauks proposed independently the *dual* primitive of *dedicated signatures*. In both notions, a signature holder can convince one or more verifiers of his knowledge of a digital signature, but cannot exploit this knowledge without being *punished* for that. In this paper, we state that a signature holder may generically provide a proof that it has a certain signature without being punished and that consequently both primitives cannot fulfill their alleged security goals. To demonstrate the feasibility of this claim, we propose the first non-interactive universal designated verifier proof of the possession of an Elgamal or a DSA signature in the random oracle model. This construction may be of independent interest.

Keywords: Universal designated verifier signatures, zero-knowledge proof, Elgamal signatures

1 Introduction

In order to control the dissemination of digital signatures, many schemes have been proposed to protect the privacy of the signer or the holder of a signature. We can cite for instance undeniable signatures [4], confirmer signatures [3] and designated verifier signatures [10]. In particular, some constructions aim at modifying a *traditional* signature into a signature with special properties (for instance a signature with controlled verification, like *universal* designated verifier signatures [19]).

⋆ This work was done while this author was a postdoctoral fellow in the Computer Security group of the Bonn/Aachen International Center for Information Technology.

In 2006, Huang, Susilo, Mu and Zhang proposed the concept of *restricted universal designated verifier signatures* while Klonowski, Kubiak, Kutylowski and Lauks proposed independently, the *dual* primitive of *dedicated signatures.* In both notions, a signature holder can convince one or more verifiers of his knowledge of a digital signature, but cannot exploit this knowledge without being *punished* for that. The main purpose of the present paper is to state that that both primitives do not achieve their purported security goals.

Background. In the electronic world, digital signatures are used to verify whether one message really comes from the alleged signer. Like handwritten signatures, standard digital signatures are *non-repudiable* and *universally verifiable.* However, universal verifiability might not suit the circumstances under which verifying signature is a valuable action.

In 1989, Chaum and van Antwerpen [4] introduced the concept of *undeniable signature* scheme in which anyone has to interact with the signer to verify the validity/invalidity of a signature. The important property of non-repudiation still holds because the signer cannot disavow a signature through a denial protocol unless the signature is indeed invalid.

Jakobsson, Sako and Impagliazzo [10] proposed *designated verifier signatures* in 1996. A designated verifier signature scheme provides authentication of a message without providing the non-repudiation property of traditional signatures and can be used to convince only one third party (*i. e.* the designated verifier and only him can be convinced about its validity or invalidity). This is due to the fact that the designated verifier can always create a signature intended for himself that is indistinguishable from an original signature.

Steinfeld, Bull, Wang and Pieprzyk [19] proposed the extended concept of *universal designated-verifier signatures* in 2003. These signatures are ordinary digital signatures with the additional functionality that any holder of a signature is able to convert it into a designated verifier signature specified to a designated verifier of his choice. Steinfeld *et al.* also showed how to construct efficient deterministic universal designated-verifier signature schemes from bilinear group-pairs and many constructions have been proposed compatible with popular digital signature schemes (*e. g.* [12, 13, 20, 21]).

In 2006, Huang, Susilo, Mu and Zhang [8] proposed the concept of *restricted universal designated verifier signature.* In this construct, a signature holder can convince up to t verifiers that he actually knows a signature, and the convincing statement is designated to these verifiers. However, when the signature holder uses the signature $t+1$ times, then the signature becomes publicly available. This primitive can potentially be used in a service such as an Internet trial-browsing service in which a user is allowed to access the service up to t times without any charge, but will be charged on the $t+1$ count of access.

In [11], Klonowski, Kubiak, Kutylowski and Lauks introduced a new kind of signatures, in a sense dual of the previous ones, that they called *dedicated* *signatures.* In their scheme, the signer can construct a signature in such a way that the recipient cannot show this signature to third parties without being punished for that. Namely, in this protocol, the signer gives the recipient a dedicated signature; the verifier derives a standard signature from it and if the verifier presents this signature to third parties, the signature together with the dedicated signature reveal the private key of the verifier. In the present paper, we prove that these two primitives are not sound since generically they cannot fulfill their alleged security goals.

Contributions of the paper. In this paper, we prove that a signature holder may generically provide a proof that it has a certain signature without being punished and that consequently both primitives are not sound. The result comes from the well-know fact that if non uniform one-way function exist then there exists a computational zero-knowledge proof system of membership for all languages having an interactive proof system of membership [6].

As a practical example to defeat the first realization of the concept of restricted universal designated verifier signatures, we remind an efficient proof of possession of a Boneh-Lynn-Shacham signature [1] (BLS signature for short) due to Hufschmitt, Lefranc and Sibert [9].

Finally, to demonstrate the feasibility of this attack on the primary dedicated signature scheme proposed by Klonowski *et al.*, we also propose the first efficient non-interactive designated verifier proof of the possession of an Elgamal signature [5] in the so called random oracle model. This result is of independant interest, as it does not exist, as far as we know, any concrete proof of knowledge of an Elgamal or a DSA signature, but a specialization of the inefficient proofs given in [15] by Nguyen, Bao, Mu and Varadharajan.

2 Definitions

In this section, we give the definition of (restricted) universal designated verifier signature scheme and dedicated signature scheme.

2.1 Notations

The set of n-bit strings is denoted by $\{0,1\}^n$ and the set of all finite binary strings (or messages) is denoted by $\{0,1\}^*$. Concatenation of two strings x and y is denoted by $x\|y$. Let $\mathcal{A}$ be a probabilistic Turing machine running in polynomial time (a PPTM, for short), and let x be an input for $\mathcal{A}$. The probability space that assigns to a string σ the probability that $\mathcal{A}$, on input x, outputs σ is denoted by $\mathcal{A}(x)$. The support of $\mathcal{A}(x)$ is denoted by $\mathcal{A}[x]$. Given a probability space S, a PPTM that samples a random element according to S is denoted by $x \stackrel{R}{\longleftarrow} S$. For a finite set X, $x \stackrel{R}{\longleftarrow} X$ denotes a PPTM that samples a random element uniformly at random from X.

2.2 Universal designated verifier signatures

In this subsection, we recall the definition of universal designated verifier signature (UDVS) schemes [19], together with the security model, that we will need to present the new Elgamal scheme in Section 3.

Definition 1. *A* universal designated verifier signature scheme Σ *is an 8-tuple*

$$\Sigma = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{VKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Designate}, \mathsf{Fake}, \mathsf{DVerify})$$

such that

- $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is a signature scheme:*
  - $\Sigma.\mathsf{Setup}$ *is a probabilistic polynomial-time Turing machine (PPTM) which takes an integer k as input. The output are the* public parameters Υ*. k is called the* security parameter*.*
  - $\Sigma.\mathsf{SKeyGen}$ *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathsf{sks}, \mathsf{pks})$ *where* sks *is called a* signing secret key *and* pks *a* signing public key*.*
  - $\Sigma.\mathsf{Sign}$ *is a PPTM which takes the public parameters, a message, and a signing secret key as inputs and outputs a bit string.*
  - $\Sigma.\mathsf{Verify}$ *is a PPTM which takes the public parameters, a message m, a bit string σ and a signing public key* pks*. It outputs a bit. If the bit output is 1 then the bit string σ is said to be a* signature *on m for* pks*.*
- $\Sigma.\mathsf{VKeyGen}$ *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathsf{skv}, \mathsf{pkv})$ *where* skv *is called a* verifying secret key *and* pkv *a* verifying public key*.*
- $\Sigma.\mathsf{Designate}$ *is a polynomial-time Turing machine (PTM) which takes the public parameters, a message m, a signing public key* pks*, a signature σ on m for* pks *and a verifying public key as inputs and outputs a bit string.*
- $\Sigma.\mathsf{Fake}$ *is a PPTM which takes the public parameters, a message, a signing public key and a verifying secret key as inputs and outputs a bit string.*
- $\Sigma.\mathsf{DVerify}$ *is a deterministic PPTM which takes the public parameters, a message m, a bit string τ, a signing public key* pks*, a verifying public key* pkv *the matching verifying secret key* skv *as inputs. It outputs a bit. If the bit output is 1 then the bit string τ is said to be a* designated verifier signature *on m from* pks *to* pkv*.*

Σ must satisfy the following properties, for all $k \in \mathbb{N}$, all $\Upsilon \in \Sigma.\mathsf{Setup}[k]$, all $(\mathsf{pks}, \mathsf{sks}) \in \Sigma.\mathsf{SKeyGen}[\Upsilon]$*, all* $(\mathsf{pkv}, \mathsf{skv}) \in \Sigma.\mathsf{VKeyGen}[\Upsilon]$ *and all messages m:*

- Correctness of Signature:

$$\forall \sigma \in \Sigma.\mathsf{Sign}[\Upsilon, m, \mathsf{sks}], \quad \Sigma.\mathsf{Verify}[\Upsilon, m, \sigma, \mathsf{pks}] = \{1\}.$$

- Correctness of Designation:

$$\forall \sigma \in \Sigma.\mathsf{Sign}[\Upsilon, m, \mathsf{sks}], \quad \forall \tau \in \Sigma.\mathsf{Designate}[\Upsilon, m, \mathsf{pks}, \sigma, \mathsf{pkv}],$$
$$\Sigma.\mathsf{DVerify}[\Upsilon, m, \tau, \mathsf{pks}, \mathsf{pkv}, \mathsf{skv}] = \{1\}.$$

– Source Hiding:

$$\Sigma.\mathsf{Designate}(\Upsilon, m, \mathsf{pks}, \Sigma.\mathsf{Sign}(\Upsilon, m, \mathsf{sks}), \mathsf{pkv}]) = \Sigma.\mathsf{Fake}(\Upsilon, m, \mathsf{pks}, \mathsf{skv}).$$

The correctness properties insure that a properly formed (designated verifier) signature is always accepted by the (designated) verifying algorithm. The source hiding property states that given a message m, a signing public key pks, a verifying public key pkv and a designated verifier signature τ on m from pks to pkv it is infeasible to determine if τ was produced by Σ.Designate or Σ.Fake.

The unforgeability notion is an extention of the classical notion of exitential unforgeability under a chosen-message attack as defined in [7].

Definition 2. *A universal designated verifier signature scheme is said strongly existentially unforgeable if no adversary (PPTM) $\mathcal{F}$ has a non-negligible advantage in the following game:*

1. *The challenger $\mathcal{C}$ takes as input a security parameter k and executes*
   $\Upsilon \leftarrow \mathsf{UDVS}.\Sigma.\mathsf{Setup}(k)$,
   $(sk_S^\star, pk_S^\star) \leftarrow \mathsf{UDVS}.\Sigma.\mathsf{KeyGen}(k, \Upsilon)$,
   $(sk_V^\star, pk_V^\star) \leftarrow \mathsf{UDVS}.\mathsf{VKeyGen}(k, \Upsilon)$.
   It gives $pk_S^\star$ and $pk_V^\star$ to the forger $\mathcal{F}$ and keeps $sk_S^\star$ and $sk_V^\star$ to itself.
2. *The forger $\mathcal{F}$ can issue the following queries:*
   i) *a signing query for some message m; the challenger $\mathcal{C}$ executes*
   $$\sigma \leftarrow \mathsf{UDVS}.\Sigma.\mathsf{Sign}(k, \Upsilon, m, sk_S^\star)$$
   and hands σ to $\mathcal{F}$;
   ii) *a verification query for pairs $(m, \tilde{\sigma})$ of his choice; $\mathcal{C}$ returns to $\mathcal{F}$ the value $\mathsf{UDVS}.\mathsf{DVerify}(k, \Upsilon, m, \tilde{\sigma}, pk, (sk_V^\star, pk_V^\star))$;*
3. *$\mathcal{F}$ outputs a V-designated verifier signature $\tilde{\sigma}^\star$ for a message $m^\star$.*

The adversary $\mathcal{F}$ succeeds if $\mathsf{UDVS}.\mathsf{DVerify}\big(k, \Upsilon, pk_S^\star, (sk_V^\star, pk_V^\star)\big) = 1$ and if $\tilde{\sigma}^\star$ has not been obtain from the signing oracle. An attacker $\mathcal{F}$ is said to $(\tau, q_s, q_v, \varepsilon)$-break the unforgeability of the UDVS scheme if he succeeds in the game within running time τ and with probability ε after having made q_s signing queries and q_v verification queries.

We will propose in Section 3 a new UDVS scheme compatible with standard Elgamal or DSA signatures, reaching these security notions. Another important security requirement concerning the undeniable signature familly is the notion of *anonymity* (roughly speaking, an anonymous designated verifier signature is indistinguishable from a random string.). Is has been precisely defined in [13] in terms of *privacy of signer's identity* for designated verifier signatures, but we will not need this property for our purpose.

2.3 Restricted universal designated verifier signatures

According to [8],

> *$\langle a \rangle$ restricted UDVS scheme is comprised of three procedures namely sign-up, designation and open. In the sign-up procedure, a user obtains a signature σ from the signer. This signature is publicly verifiable, but this is kept by the user (and hence, the signature holder) to be used in the designation procedure. $\langle \dots \rangle$ In the designation procedure, the signature holder can designate the signature to up to t verifiers. The verifiers will be convinced with the authenticity of the signature, but they cannot convince any other third party about this fact. When the signature holder uses the signature to convince the $t+1$ verifier, the open procedure can be invoked to reveal the signature $\langle \dots \rangle$.*

One of our purpose in this paper is to disprove the relevancy of this primitive. Therefore, it is necessary to describe formally this primitive.

Definition 3. *Let t be a positive integer. A t*-restricted universal designated verifier signature scheme Σ *is a 9-tuple*

$$\Sigma = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{VKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Designate}, \mathsf{Fake}, \mathsf{DVerify}, \mathsf{Open})$$

such that

- $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{VKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Designate}, \mathsf{Fake}, \mathsf{DVerify})$ *is a universal designated signature scheme;*
- $\Sigma.\mathsf{Open}$ *is a PPTM which takes the public parameters, a message, $t+1$ bit strings, a signing public key and $t+1$ verifying public keys and outputs a bit string.*

Σ must satisfy the following property for all messages m:

- Opening:

$$\begin{gathered} \forall k \in \mathbb{N}, \forall \Upsilon \in \Sigma.\mathsf{Setup}[k], \forall (\mathsf{pks}, \mathsf{sks}) \in \Sigma.\mathsf{SKeyGen}[\Upsilon], \forall \sigma \in \Sigma.\mathsf{Sign}[\Upsilon, m, \mathsf{sks}], \\ \forall (\mathbf{pkv}, \mathbf{skv}) = [(\mathsf{pkv}_1, \mathsf{skv}_1), \dots, (\mathsf{pkv}_{t+1}, \mathsf{skv}_{t+1})] \in \Sigma.\mathsf{VKeyGen}[\Upsilon]^{t+1} \\ \forall \tau_1 \in \Sigma.\mathsf{Designate}[\Upsilon, m, \mathsf{pks}, \sigma, \mathsf{pkv}_1], \\ \dots \\ \forall \tau_{t+1} \in \Sigma.\mathsf{Designate}[\Upsilon, m, \mathsf{pks}, \sigma, \mathsf{pkv}_{t+1}] \\ \#\{\tau_1, \dots, \tau_{t+1}\} = t+1 \Rightarrow \Sigma.\mathsf{Open}[\Upsilon, m, \tau_1, \dots, \tau_{t+1}, \mathbf{pkv}, \mathsf{pks}] = \{\sigma\}. \end{gathered}$$

This property formalizes the (erroneous, as we will see) fact that when the signature is designated to convince $t+1$ verifiers, the open procedure can be invoked to reveal the original signature.

In [8], Huang *et al.* presented a restricted UDVS scheme based on BLS signatures.

2.4 Dedicated signatures

The definition of dedicated signatures is not given explicitly and formally, but can be readily easily deduced from this informal description from [11]. We give here a formal definition (although the concept is pointless as we will see).

Definition 4. *A* dedicated signature scheme Σ *is an 7-tuple*

$$\Sigma = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{VKeyGen}, \mathsf{DDSCreate}, \mathsf{Retrieve}, \mathsf{Verify}, \mathsf{Punish})$$

such that

- $\Sigma.\mathsf{Setup}$ *is a PPTM which takes an integer k as input. The output are the* public parameters Υ. *k is called the* security parameter.
- $\Sigma.\mathsf{SKeyGen}$ *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathsf{sks}, \mathsf{pks})$ *where* sks *is called a* signing secret key *and* pks *a* signing public key.
- $\Sigma.\mathsf{VKeyGen}$ *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathsf{skv}, \mathsf{pkv})$ *where* skv *is called a* verifying secret key *and* pkv *a* verifying public key.
- $\Sigma.\mathsf{DDSCreate}$ *is a PPTM which takes the public parameters, a message, a signing secret key and a verifying public key as inputs and outputs a bit string.*
- $\Sigma.\mathsf{Retrieve}$ *is a PPTM which takes as input the public parameters, a message, a bit string, a signing public key and a verifying secret key as input. The output is a bitstring.*
- $\Sigma.\mathsf{Verify}$ *is a PTM which takes the public parameters, a message m, a bit string σ and a signing public key* pks. *It outputs a bit. If the bit output is 1 then the bit string σ is said to be a* signature *on m for* pks.
- $\Sigma.\mathsf{Punish}$ *is a PPTM which takes the public parameters, a message, two bit strings, a signing public key and a verifying public key and outputs a bit string.*

Σ must satisfy the following properties, for all $k \in \mathbb{N}$, all $\Upsilon \in \Sigma.\mathsf{Setup}[k]$, all $(\mathsf{pks}, \mathsf{sks}) \in \Sigma.\mathsf{SKeyGen}[\Upsilon]$, all $(\mathsf{pkv}, \mathsf{skv}) \in \Sigma.\mathsf{VKeyGen}[\Upsilon]$ and all messages m:

- Correctness of Retrieval:

$$\forall \tau \in \Sigma.\mathsf{DDSCreate}(\Upsilon, m, \mathsf{sks}, \mathsf{pkv}), \quad \forall \sigma \in \Sigma.\mathsf{Retrieve}(\Upsilon, m, \tau, \mathsf{pks}, \mathsf{skv}),$$
$$\Sigma.\mathsf{Verify}[\Upsilon, m, \sigma, \mathsf{pks}] = \{1\}.$$

- Punishement:

$$\forall \tau \in \Sigma.\mathsf{DDSCreate}(\Upsilon, m, \mathsf{sks}, \mathsf{pkv}), \quad \forall \sigma \in \Sigma.\mathsf{Retrieve}(\Upsilon, m, \tau, \mathsf{pks}, \mathsf{skv}),$$
$$\Sigma.\mathsf{Punish}[\Upsilon, m, \tau, \sigma, \mathsf{pks}, \mathsf{pkv}] = \{\mathsf{skv}\}.$$

The first property can be restated as follows: $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is a signature scheme, where Sign is the PPTM which takes the public parameters, a message m, a signing key pair $(\mathsf{sks}, \mathsf{pks})$ and a verifying key pair $(\mathsf{skv}, \mathsf{pkv})$

as inputs, obtains τ by executing $\Sigma.\mathsf{DDSCreate}(\Upsilon, m, \mathsf{sks}, \mathsf{pkv})$ and returns σ obtained by executing $\Sigma.\mathsf{Retrieve}(\Upsilon, m, \tau, \mathsf{pks}, \mathsf{skv})$. The second property means that the verifier loses his own secrets when exhibiting the signature.

In [11], Klonowski *et al.* proposed a realization of this primitive based on Elgamal signatures. It is well known, that the original Elgamal signature scheme is existentially forgeable (see [17] for instance). As in [11], we consider only, in this paper, the variant of Elgamal scheme where a collision-resistant hash function is applied to the message to be signed and only the digest is actually "signed". This "hashed Elgamal signatures" has not been proven existentially unforgeable but there exist arguments in favor of their security in the so-called *generic group model* [2].

It is worth noting that this scheme presented is very similar to the anonymous designated verifier signature scheme proposed in 2003 by Saeednia, Kremer and Markowitch [18].

3 Universal designated verifier Elgamal and DSA signatures

3.1 Description of the scheme

We present in this section the first efficient universal designated verifier Elgamal and DSA signatures. We describe our new scheme with the Elgamal signature for simplicity. It works as follows:

- $\mathsf{UDVS}.\Sigma.\mathsf{Setup}$: public parameters include the output of a DL-parameter-generator as well as an integer n, a collision-resistant hash function $h : \{0,1\}^* \to \mathbb{Z}_p^*$: $\Upsilon := \{n, p, \mathbb{Z}_p^*, g, h\}$.
- $\mathsf{UDVS}.\Sigma.\mathsf{SKeyGen}$: a signer's private key is a randomly chosen $a \xleftarrow{R} \mathbb{Z}_p^*$; his public key consists of a group element $y_A = g^a \mod p$.
- $\mathsf{UDVS}.\Sigma.\mathsf{Sign}$: given a message $m \in \{0,1\}^*$, the signer picks $k \xleftarrow{R} \mathbb{Z}_p^*$ and sets $u = g^k \mod p$ and $v = k^{-1}(h(m) - au) \mod p-1$. The signature is $\sigma = (u, v)$.
- $\mathsf{UDVS}.\Sigma.\mathsf{Verify}$: a plain signature $\sigma = (u, v)$ on m is accepted if $y_A^u u^v = g^{h(m)} \mod p$.
- $\mathsf{UDVS}.\mathsf{VKeyGen}$: a designated verifier's private key is a random element $b \xleftarrow{R} \mathbb{Z}_p^*$; the matching public key is $y_B = g^b \mod p$.
- $\mathsf{UDVS}.\mathsf{Designate}$: the holder of a signature $\sigma = (u, v)$, who chooses B as designated verifier produces the designated verifier signature $\tau = (u, \mathcal{P})$ where $\mathcal{P}$ is the non-interactive zero-knowledge proof of knowledge of one discrete logarithm among two $\mathsf{HVZKPK}\left(\log_u(g^{h(m)} y_A^{-u}) \vee \log_g(y_B)\right)$, derived from the interactive the proof described in Fig. 1.
  More precisely, the holder of the signature σ computes $e = h(c_1||c_2||m||u)$, with $y_1 = u$, $y_2 = g$, $\alpha_1 = g^{h(m)} y_A^{-u} \mod p$ and $\alpha_2 = y_B$. Then τ is the 5-tuple (u, e_1, e_2, s_1, s_2).

Prover		Verifier
$r_1, e_2, s_2 \in_R \mathbb{Z}_p^*$		
$c_1 = y_1^{r_1} \mod p$		
$c_2 = \alpha_2^{e_2} y_2^{s_2} \mod p$		
	$\xrightarrow{c_1, c_2}$	
		$e \in_R \mathbb{Z}_p^*$
	$\xleftarrow{e}$	
$e_1 = e_2 - e \mod p-1$		
$s_1 = r_1 - e_1 x_1 \mod p-1$		
	$\xrightarrow{(e_1, s_1), (e_2, s_2)}$	
		$c_1 \stackrel{?}{=} \alpha_1^{e_1} y_1^{s_1} \mod p$
		$c_2 \stackrel{?}{=} \alpha_2^{e_2} y_1^{s_2} \mod p$
		$e \stackrel{?}{=} e_1 + e_2 \mod p-1$

The prover knows $x_1 = \log_{y_1}(\alpha_1)$.

Fig. 1. HVZKPK $\left(\log_{y_1}(\alpha_1) \vee \log_{y_2}(\alpha_2)\right)$

- UDVS.Fake: the designated verifier produces the designated verifier signature $\tau = (u, \mathcal{P})$ where u is a random element $u \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and $\mathcal{P}$ is the non-interactive variant of the proof HVZKPK $\left(\log_u(g^{h(m)} y_A^{-u}) \vee \log_g(y_B)\right)$ (performed thanks to the knowledge of $\log_g(y_B)$).
- UDVS.DVerify: given a purported signature $\tau = (u, \mathcal{P})$, the designated verifier checks the correctness of the non-interactive proof $\mathcal{P} = (e_1, e_2, s_1, s_2)$ by computing $c_1 = (g^{h(m)} y_A^{-u})^{e_1} u^{s_1}$ and $c_2 = y_B^{e_2} g^{s_2}$ and checking that $h(c_1 || c_2 || m || u) = e_1 + e_2 \mod q$.

DSA signatures. The difference between Elgamal signatures and DSA signatures lies essentially in the fact that computation are done modulo a prime q dividing $p-1$ instead of working modulo $p-1$. Therefore, the technique previously described can be trivially adapted in the case of the DSA signatures. The resulting scheme is also the first universal designated verifier signature scheme based on DSA.

Remark 1. A UDVS scheme reaches the source-hiding property if and only if the designated verifier can produce one bitstring that is indistinguishable from one that was generated by the signature holder. In the Elgamal-based scheme, that we have described this property is fulfilled. However, it remains an open problem to design an efficient UDVS scheme compatible with Elgamal/DSA signatures with the stronger property that the designated verifier can always produce identically distributed transcripts that are indistinguishable from the one generated by the signature holder:

$$\forall k \in \mathbb{N}, \forall \Upsilon \in \Sigma.\mathsf{Setup}[k], \forall(\mathsf{pks},\mathsf{sks}) \in \Sigma.\mathsf{SKeyGen}[\Upsilon],$$
$$\forall(\mathsf{pkv},\mathsf{skv}) \in \Sigma.\mathsf{VKeyGen}[\Upsilon], \forall m \in \{0,1\}$$
$$\left(\tau \xleftarrow{R} \Sigma.\mathsf{Sign}(\Upsilon, m, \mathsf{sks}); \Sigma.\mathsf{Designate}(\Upsilon, m, \mathsf{pks}, \tau, \mathsf{pkv}])\right) = \Sigma.\mathsf{Fake}(\Upsilon, m, \mathsf{pks}, \mathsf{skv}).$$

3.2 Security analysis

Theorem 1 (Unforgeability). *Let $\mathcal{F}$ be a forger that $(t, q_s, q_v, \varepsilon)$-breaks the Elgamal UDVS scheme in the q_h-random oracle model, with $\varepsilon > 7q_H/2^k$. There exists a $(t', \lceil(14q_H + 2)q_s/\varepsilon\rceil, \varepsilon')$-EF-CMA algorithm $\mathcal{A}$ against hashed Elgamal signature such that $\varepsilon' \geq 1/9$ after running $\mathcal{F}$ by $\left\lceil \frac{2}{\varepsilon} + \frac{14q_h}{\varepsilon} \right\rceil$ times.*

Proof. The proof relies on the well-known forking technique proposed in 1996 by Pointcheval and Stern [17]: assuming that an attacker can forge a designated verifier signature, another algorithm could obtain, by replaying sufficiently many times this attacker with randomly chosen hash functions (i.e. random oracles), two forged designated verifier signatures of the same message and with the same randomness. Then, these two forged signatures could be used to solve some computational problem which is assumed to be intractable: producing an Elgamal forgery or computing a discrete logarithm.

The algorithm $\mathcal{A}$ takes as inputs the public parameters Υ, an Elgamal signing public-key $\mathsf{pks} = y$ and has access to a signing oracle $\mathfrak{S}_A$ that it can query up to $\lceil(14q_H+2)q_S/\varepsilon\rceil$ times. It produces a verifying public key pkv in such a way that if it obtains in the simulation the discrete logarithm of pkv in base g, then it will also know the discrete logarithm of pks in base g and therefore can readily forge a signature (*i. e.* $\mathsf{pkv} = y^r$ for a known $r \in [\![1, p-1]\!]$ picked uniformly at random). The adversary $\mathcal{A}$ executes the forger $\mathcal{F}$ many times (at most $\lceil(14q_h + 2)\varepsilon^{-1}\rceil$) on the entries $(\Upsilon, \mathsf{pks}, \mathsf{pkv})$ with different random tapes and/or random oracles. The algorithm $\mathcal{A}$ will run in two stages.

First stage: The forger $\mathcal{F}$, with random tape ϖ, can make q_S queries to the signing oracle $\mathfrak{S}_F$ and q_H queries to the random oracle $\mathcal{H}$. We denote by $(e_1, \ldots, e_{q_H})$ the list of the q_H answers of the random oracle. Therefore, we can see a random choice of the random oracle as a random choice of such a vector $\boldsymbol{e}$. In his first stage, $\mathcal{A}$ executes the forger $\mathcal{F}$ at most $(2/\varepsilon)$ times with different random tapes and random oracles, until it outputs a valid forgery. For each execution, the queries made by $\mathcal{F}$ to the signing oracle $\mathfrak{S}_F$ are simply transfered to $\mathfrak{S}_A$ and the returned signatures are stored by $\mathcal{A}$.

Eventually, in one execution $\mathcal{F}$ outputs a forged message/signature pair $(m^\star, \tau^\star)$ where $\tau^\star = (u^\star, e_1^\star, e_2^\star, s_1^\star, s_2^\star)$ under the public keys $(\mathsf{pks}, \mathsf{pkv})$. The probability that $\mathcal{F}$ outputs a valid forgery at the end of one execution is ε. Since $\mathcal{H}$ is a random oracle, the probability that $\mathcal{F}$ succeeds and has not submitted $(c_1^\star || c_2^\star || m^\star || u^\star)$ (where $c_1^\star = (g^{h(m)} y_A^{-u^\star})^{e_1^\star} u^{s_1^\star}$ and $c_2^\star = y_B^{e_2^\star} g^{s_2^\star}$) to $\mathcal{H}$ is less than

2^{-k}. Therefore, the probability that $\mathcal{F}$ returns a forged message/signature which has been queried to the random oracle in one execution is at least $6\varepsilon/7$. We define $\mathrm{Ind}(\varpi, \boldsymbol{e})$ to be the index of this query. We then define the sets

$$\mathcal{S}_i = \{(\varpi, \boldsymbol{e}) | \mathcal{F}^{\mathfrak{S}_F, \boldsymbol{e}}(\varpi) \text{ succeeds \& } \mathrm{Ind}(\varpi, \boldsymbol{e}) = i\}$$

for $i \in [\![1, q_H]\!]$ and $\mathcal{S} = \bigcup_{i=1}^{q_H} \mathcal{S}_i$. We have $\Pr[\mathcal{S}] \geq 6\varepsilon/7$.

Therefore, $\mathcal{A}$ at the end of the first stage will get at least one pair

$$(\varpi, \boldsymbol{e} = (e_1, \ldots, e_{q_H}))$$

(and a corresponding list of signatures) leading to a forgery with probability at least $1 - \exp(-12/7) \geq 4/5$ after having queried $\mathfrak{S}_A$ at most $\lceil 2q_S/\varepsilon \rceil$ times. Let us denote $\gamma = \mathrm{Ind}(\varpi, \boldsymbol{e})$ the index of query of the forged message to the random oracle. Let I be the set consisting of the most likely indices i:

$$I = \{i \in [\![1, q_H]\!], \Pr[\mathcal{S}_i | \mathcal{S}] \geq 1/2q_H\}.$$

It is easy to see [17, Lemma 3], that in case of success, the index γ lies in I with probability at least 1/2. Moreover, with probability greater than 1/5 the pair $(\varpi, \boldsymbol{e})$ belongs to S_γ.

Second stage: In the second stage, $\mathcal{A}$ executes $\mathcal{F}$ with the random tape ϖ which led to the forgery in the first step and different random oracles $\boldsymbol{e'} = (e'_1, \ldots, e'_{q_H})$ such that $e_i = e'_i$ for $i < \gamma$, e'_γ is picked uniformly at random in $[\![1, p-1]\!] \setminus \{e_\gamma\}$ and the e'_j's are picked uniformly at random in $[\![1, p-1]\!]$ for $j > \gamma$. $\mathcal{A}$ uses the signatures recorded in the first step to answer F signature queries before the γ-th random oracle query and its own signing oracle $\mathfrak{S}_A$ after that point.

After running $\mathcal{F}$ $\lceil 14q_H/\varepsilon \rceil$ times, $\mathcal{A}$ will obtain, if $(\varpi, \boldsymbol{e}) \in S_\gamma$, with probability at least 3/5 (see [17, Lemma 1] for instance), two valid designated verifier signatures on $m^\star$: $(u^\star, e_1^\star, e_2^\star, s_1^\star, s_2^\star)$ and $(u^\star, e_1^\dagger, e_2^\dagger, s_1^\dagger, s_2^\dagger)$ such that

$$e_1^\star + e_2^\star \neq e_1^\dagger + e_2^\dagger \mod p$$

and

$$(g^{h(m)} y_A^{-u^\star})^{e_1^\star} u^{s_1^\star} = (g^{h(m)} y_A^{-u^\star})^{e_1^\dagger} u^{s_1^\dagger} \mod p \text{ and } y_B^{e_2^\star} g^{s_2^\star} = y_B^{e_2^\dagger} g^{s_2^\dagger} \mod p.$$

If $e_1^\star \neq e_1^\dagger \mod p$, then $\mathcal{A}$ can readily obtain a valid Elgamal signature on $m^\star$ as $(u^\star, v^\star)$ where

$$v^\star = (s_1^\star - s_1^\dagger)/(e_1^\star - e_1^\dagger) \mod p.$$

Similarly, $e_2^\star \neq e_2^\dagger \mod p$, then $\mathcal{A}$ can retrieve the discrete logarithm of pkv in base g, and therefore the discrete logarithm of pks in base g and can easily produce a Elgamal signature on the message of his choice. □

The complexity assumption used in the previous theorem is the best we can hope for. Indeed, if there exists a polynomial time EF-CMA adversary against hashed Elgamal signature then there exists a forger that breaks the Elgamal UDVS scheme with the same advantage in the same running time.

The results from [16] show that it is very unlikely that the existential unforgeability of hashed Elgamal signatures can be reduced to the discrete logarithm problem in the standard security model. However, it is worth noting that the hashed Elgamal signature scheme is a special case of the protocol AbstractDSA which has been proposed and analyzed[3] by Brown in 2005 [2]. Let us recall the *Theorem 3* from [2] which asserts the existential unforgeability of the scheme in the generic group model:

Theorem 2 ([2]). *If there exists an $(\epsilon_F, \tau_F, q_F)$-forger $\mathcal{F}$ of AbstractDSA in the generic group model with uniform hash function h and an almost-invertible conversion function f in the generic group model for A_n, then there exists an (ϵ_C, τ_C)-collision-finder C_h and (ϵ_Z, τ_Z)-zero-finder where*

$$\epsilon_C + \epsilon_Z \geq \epsilon_F + \frac{\tau_F^2}{2n} \text{ and } \tau_C, \tau_Z \leq 2\tau_F.$$

For the specific instance of AbstractDSA investigated in this paper, the conversion function is the identity map which is trivially almost invertible [2, § 2.2.2] and therefore it can be argued that this theorem applies to hashed Elgamal signatures.

4 On the weakness of restricted universal designated verifier and dedicated signatures

As mentioned above, the dedicated verifier may provide a zero-knowledge proof that it has a certain signature without presenting it. In many legal systems it would suffice in the court to present such a proof in order to derive legal consequences of the signed document.

The weakness and eventually the meaningless of these two concepts of signatures comes from the following theorem. It is among the most important results in zero-knowledge protocols and state the possible construction, using commitment schemes, of a zero-knowledge proof system for all languages in NP. This theorem was proved in [6] by Goldreich, Micali and Widgerson.

Theorem 3. *If non-uniform one-way functions exist, then there exists a computational zero-knowledge proof system of membership for all languages having an interactive proof system of membership.*

The impact of the theorem on the two kinds of signature is described below.

[3] We refer to [2] for the definitions of uniform hash function, almost-invertible conversion function and zero-finder adversary.

4.1 The case of restricted UDVS: proving the possession of a BLS signature

In Huang *et al.* paper, the authors propose a pairing-based scheme. A user receive a BLS signature, and should convince only a given number of third parties of the validity of this signature. We argue that these restriction is not possible since the signature holder can always prove the validity of the BLS signature he holds by applying the zero-knowledge proof of the possession of a BLS signature described in Fig. 2.

Prover		Verifier
$r \in_R \llbracket 1, q-1 \rrbracket$ $w = e(P,P)^r$		
	$\xrightarrow{\quad w \quad}$	
		$c \in_R \llbracket 0, q-1 \rrbracket$
	$\xleftarrow{\quad c \quad}$	
$Y = rP + c\sigma$		
	$\xrightarrow{\quad Y \quad}$	
		$e(P,Y) \stackrel{?}{=} w \times e(aP, H(m))^c$

Prover: $(pk, sk) = (aP, a)$ and $\sigma = aH(m)$

Fig. 2. Proof of knowledge of a BLS signature [9]

4.2 The case of dedicated signatures: proving the possession of an Elgamal signature

In Klonowski *et al.* paper [11], the authors propose a scheme based on Elgamal signatures. In this scheme, the signer is suppose to construct his signature in such a way that the recipient cannot show the signature to third parties without being punished (for instance, his secret key is revealed). The sent signature is called a dedicated signature, but the recipient can derive a true Elgamal signature from it. Therefore once the holder has computed this Elgamal signature, he can use our new protocol defined in Section 3 to convince *any* third party without being punished. Indeed, the signature holder can transform the Elgamal signature into a signature designated to the third party.

5 Conclusion

Signatures of knowledge allow a prover to prove the knowledge of a secret with respect to some public information noninteractively. In our case, this helps to defeat some concepts which aim at controlling the holder of a signature. To this purpose, we propose for the two concepts of restricted universald designated verifier signatures and dedicated signatures two proofs which permit the signature holder to prove his knowledge of a signature without being troubled. The first one is a classical proof of a BLS signature. The second one is the first efficient designated verifier proof of an Elgamal signature.

An interesting open problem would be to have an efficient zero-knowledge proof of the possession of an Elgamal (or DSA) signature. What we proposed is an efficient designated verifier proof, which might not be relevant in some special cases. Moreover, the concept of punishing a signature holder who unauthorizedly disclose a signature is still open. The presence of an authority seems necessary to design such a scheme.

References

1. D. Boneh, H. Shacham, and B. Lynn, *Short signatures from the Weil pairing.*, J. of Cryptology, 17(4), 2004, pp. 297–319.
2. D. R. L. Brown, *Generic Groups, Collision Resistance, and ECDSA* Des. Codes Cryptography 35(1), 2005, pp. 119–152
3. D. Chaum, *Designated Confirmer Signatures.*, Advances in Crypology - Eurocrypt'94 (A. De Santis, Ed.), Lect. Notes Comput. Sci., vol. 950, Springer, 1995, pp. 86–9
4. D. Chaum and H. van Antwerpen, *Undeniable Signatures.*, Advances in Cryptology - Crypto'89 (G. Brassard, ed.), Lect. Notes Comput. Sci., vol. 435, Springer, 1990, pp. 212–216.
5. T. Elgamal, *A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.*, Advances in Cryptology, Crypto'84, (G. R. Blakley, David Chaum, eds.), Lect. Notes Comput. Sci., vol. 196 , Springer, 1985, pp. 10–18.
6. O. Goldreich, S. Micali, A. Wigderson, *Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems.* J. ACM, 38(3), 1991, pp. 691–729
7. S. Goldwasser, S. Micali, R. L. Rives,*A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks.*, SIAM J. Comput., 1(2), 1988, pp. 281–308
8. X. Huang, W. Susilo, Y. Mu, and F. Zhang, *Restricted Universal Designated Verifier Signature*, Ubiquitous Intelligence and Computing, Third International Conference, UIC 2006 (J. Ma, H. Jin, L. T. Yang, and J. J. P. Tsai, eds.), Lect. Notes Comput. Sci., vol. 4159, Springer, 2006, pp. 874–882.
9. E. Hufschmitt, D. Lefranc, and H. Sibert, *A Zero-Knowledge Identification Scheme in Gap Diffie-Hellman Groups.*, preprint, 2005.
10. M. Jakobsson, K. Sako, and R. Impagliazzo, *Designated Verifier Proofs and Their Applications.*, Advances in Cryptology - Eurocrypt'96 (U. M. Maurer, ed.), Lect. Notes Comput. Sci., vol. 1070, Springer, 1996, pp. 143–154.
11. M. Klonowski, P. Kubiak, M. Kutylowski, and A. Lauks, *How to Protect a Signature from Being Shown to a Third Party.* , Trust and Privacy in Digital Business, Third International Conference, TrustBus 2006 (S. Fischer-Hübner, S. Furnell, and C. Lambrinoudakis, eds.), Lect. Notes Comput. Sci., vol. 4083, Springer, 2006, pp. 192–202.

12. F. Laguillaumie, B. Libert, and J.-J. Quisquater, *Universal Designated Verifier Signatures Without Random Oracles or Non-Black Box Assumptions.*, Fifth Conference on Security and Cryptography for Networks, SCN'06 (R. de Prisco and M. Yung, eds.), Lect. Notes Comput. Sci., vol. 4116, Springer, 2006, pp. 63–77.
13. F. Laguillaumie and D. Vergnaud, *Designated Verifier Signatures: Anonymity and Efficient Construction from* any *Bilinear Map.*, Fourth Conference on Security in Communication Networks, SCN 2004 (C. Blundo and S. Cimato, eds.), Lect. Notes Comput. Sci., vol. 3352, Springer, 2005, pp. 107–121.
14. H. Lipmaa, G. Wang and F. Bao, *Designated Verifier Signature Schemes: Attacks, New Security Notions and A New Construction.*, 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005 (L. Caires and L. Monteiro, eds.), Lect. Notes Comput. Sci., vol. 3580, Springer, 2005, pp. 61–71.
15. K. Q. Nguyen, F. Bao, Y. Mu and V. Varadharajan, *Zero-Knowledge Proofs of Possession of Digital Signatures and its Applications*, Information and Communication Security, Second International Conference, ICICS'99, (V. Varadharajan, Y. Mu, eds.), Lect. Notes Comput. Sci., vol. 1726, Springer, 1999, pp. 103–118.
16. P. Paillier and D. Vergnaud, *Discrete-Log Based Signatures May Not Be Equivalent to Discrete-Log.*, Advances in Cryptology - Asiacrypt 2005 (B. Roy, ed.), Lect. Notes Comput. Sci., vol. 3788, Springer, 2005, pp. 1–20.
17. D. Pointcheval and J. Stern, *Security Arguments for Digital Signatures and Blind Signatures.* J. Cryptology, 13(3), 2000, pp. 361–396
18. S. Saeednia, S. Kremer, and O. Markowitch, *An Efficient Strong Designated Verifier Signature Scheme.*, Information Security and Cryptology - ICISC 2003, Sixth International Conference (J. I. Lim and D. H. Lee, eds.), Lect. Notes Comput. Sci., vol. 2971, Springer, 2004, pp. 40–54.
19. R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk, *Universal Designated-Verifier Signatures.*, Advances in Cryptology - Asiacrypt 2003 (C.-S. Laih, ed.), Lect. Notes Comput. Sci., vol. 2894, Springer, 2003, pp. 523–542.
20. R. Steinfeld, H. Wang, and J. Pieprzyk, *Efficient Extension of Standard Schnorr/RSA signatures into Universal Designated-Verifier Signatures.*, 7th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2004 (F. Bao, R. H. Deng, and J. Zhou, eds.), Lect. Notes Comput. Sci., vol. 2947, Springer, 2004, pp. 86–100.
21. D. Vergnaud, *New Extensions of Pairing-based Short Signatures into Universal Designated Verifier Signatures.*, 33rd International Colloquium on Automata, Languages and Programming, ICALP 2006 (M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, eds.), Lect. Notes Comput. Sci., vol. 4052, Springer, 2006, pp. 58–69.