An Efficient Strong Designated Verifier Signature Scheme

Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch

Université Libre de Bruxelles
Département d'Informatique
Bd du Triomphe - CP212
1050 Bruxelles, Belgium
{saeednia,skremer,omarkow}@ulb.ac.be

Abstract. This paper proposes a designated verifier signature scheme based on the Schnorr signature and the Zheng signcryption schemes. One of the advantages of the new scheme compared with all previously proposed schemes is that it achieves the "strong designated verifier" property without encrypting any part of the signatures. This is because the designated verifier's secret key is involved in the verification phase. Another advantage of the proposed scheme is the low communication and computational cost. Generating a signature requires only one modular exponentiation, while this amount is two for the verification. Also, a signature in our scheme is more than four times shorter than those of known designated verifier schemes.

Key words: Signature, Designated verifier, Signcryption, Discrete logarithm.

1 Introduction

In 1989, Chaum and van Antwerpen [4] introduced the notion of undeniable signatures with the goal of enabling signers to have complete control over their signatures. That is, the verification of such signatures requires the participation of the signer (by means of an interactive protocol), in order to avoid undesirable verifiers getting convinced of the validity of the signatures. However, these signatures do not always achieve their goal, because of blackmailing [6, 7] and mafia attacks [5]. The problem is due to the fact that the signer does not know to whom s/he is proving the validity of a signature.

This weakness of undeniable signatures motivated the parallel introduction of designated verifier signatures by Jakobsson, Sako and Impagliazzo [8], as well as private signatures by Chaum [3]. Both of these signatures are based on the same idea and are nearly identical. In the rest of this paper, we only focus on the Jakobsson et al. scheme as it resulted in an academic publication. Designated verifier signatures provide authentication of a message, without however having the non-repudiation property of traditional signatures. In other words, they convince one—and only one—specified recipient that they are valid, but unlike standard digital signatures, nobody else can be convinced about their validity or invalidity. The reason is that the designated verifier in these schemes is able to create a signature intended to himself that is indistinguishable from a "real" signature. Therefore, when Bob receives a signature from Alice, he will certainly trust that it is originated from Alice upon verifying it, since he knows that he has not generated it himself. However, another party, Cindy, has no reason to accept such a signature as Alice's one, because she knows that Bob is fully capable to produce it himself.

Designated verifier signatures are very useful in various situations where the signer of a message should be able to specify who may be convinced by his/her signature. Let us consider the following example.

Suppose that a public institution initiates a call for tenders, asking some companies to propose their prices for a set of instruments and tasks to be accomplished. The institution may require the companies to sign their offers in order to make sure that they are actually authentic and originated from whom they claim to be. This is a valid requirement, but no company involved in this process desires its offer to affect other tenderers' decisions. That is, a company may capture a competitor's signed offer on the transmission line (to the institution) and prepares its offer consequently in order to increase its chance to be selected by the institution.

To prevent this, the companies may obviously encrypt their offers and signatures in order that they may only be read and verified by the institution. But, nothing prevents the latter to reveal them once decrypted. Indeed, since the institution's goal is to obtain a good price (as low as possible), it could show some signed offers to some other companies to influence them in making "good" offers.

So, the here raised question is about the conflict between authenticity and privacy. Designated verifier signatures are a solution to this problem. With such signatures, while the institution is convinced about the origin and the authenticity of an offer, it cannot transfer this conviction to others.

In 1996, Jakobsson et al. [8] proposed a designated verifier signature scheme as a non-interactive designated verifier proof of undeniable signatures (see section 3 for a description and the appendix for our cryptanalysis of this scheme). More recently, in 2001, Rivest, Shamir and Tauman introduced ring signatures [11], allowing to generate a signature linked to a group of potential signers. A special case of these signatures (by setting the size of the group to two) provides designated verifier signatures (see section 3).

Although these schemes are *signer ambiguous*, in the sense that one cannot verify whether the signer or the designated verifier issued the signature, they remain universally verifiable, i.e. anybody can make sure that there are only two potential signers. Hence, considering again the example above, if the companies' offers are sent just being signed using these designated verifier schemes, the signatures may be captured on the line before arriving at the institution, so that one can identify the signer, since it is now sure that the institution did not forge the signature. As indicated before, one possible solution, that is however expensive in terms of computational cost, is to encrypt each signature with the designated verifier's public key. This stronger requirement, called *strong designated verifier*, was briefly discussed in [8].

In this paper, we introduce a new efficient designated verifier signature scheme that is based on a combination of the Schnorr signature [12] and the Zheng signcryption schemes [13]. It requires only 1 modular exponentiation to generate and 2 modular exponentiations to verify a signature, i.e. no additional exponentiations are needed to convert the original Schnorr or Zheng schemes into a designated verifier signature scheme. Moreover, our scheme directly provides the strongness property without requiring any encryption of the signatures. This is particularly interesting in terms of computational cost as we will see further in this paper. Finally, the signatures in our scheme are very small in size.

The paper is organised as follows. In section 2, we recall definitions of designated verifier proofs of [8] and we give new definitions that we believe more suitable for our further analysis. In section 3, we briefly recall the existing designated verifier signature schemes. Section 4 presents the new scheme and section 5 discusses its security. In section 6, we consider the strong designated verifier property of our scheme and in section 7 we see a comparison with other designated verifier schemes in terms of performance. Finally, we show how to break and repair the Jakobsson et al. scheme in the appendix.

2 Definitions

Our goal is to allow Alice proving the validity of a statement Ω to Bob in such a way that, while Bob is convinced of this fact, he cannot transfer this conviction to other people.

As suggested in [8], when Alice wants to convince Bob—and only Bob—of the truth of the statement Ω, she should prove the statement "$\Omega \vee$ I know Bob's secret key". Bob, who is aware that he has not prepared the proof himself and knows that Alice does not know his secret key, will accept the validity of the first part of the statement (i.e., Ω) while no other verifier will be able to decide which part of the disjunction is true.

Informal definitions of the designated verifier proofs are given in [8]. We believe that these definitions, though completely persuasive, do not fully capture our intuition of the designated verifier proofs. Hereafter, we recall them and give more intuitive definitions that would be more suitable for further analysis of such schemes.

Definition 1 (Designated Verifier).
Let (P_A, P_B) be a protocol for Alice to prove the truth of the statement Ω to Bob. We say that Bob is a designated verifier if for any protocol (P_A, P'_B, P_C) involving Alice, Bob and Cindy, by 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).

This definition clearly tells us that if Bob, after having received a proof (signature) from Alice, has a way to prove to Cindy the truth of a given statement, then he can produce indistinguishable transcripts by his own. As a consequence, whatever Bob can do with the "real" transcripts, he will be able to do with the "simulated" transcripts as well. Thus, Cindy being aware of this fact, will never be convinced by Bob's proof, whatever the protocol that Bob initiates.

Put in more formal words, we can define designated verifier proofs as follows:

Definition 2 (New Designated Verifier).
Let $P(A, B)$ be a protocol for Alice to prove the truth of the statement Ω to Bob. We say that Bob is a designated verifier if he can produce identically distributed transcripts that are indistinguishable from those of $P(A, B)$.

This is very close to the definition of zero-knowledge proofs by means of a simulator, except that a designated verifier proof is simulable with "no rewinding" of the simulator and especially with any challenge size. This precisely means that, in terms of signatures, the designated verifier signatures are the only kind of signature schemes that are zero-knowledge for the intended verifier. No classical signature scheme providing non-repudiation, including those derived from zero-knowledge identification schemes by means of a hash function, may be zero-knowledge, otherwise it is insecure.

Strong designated verifier proofs. In some circumstances, Cindy may be convinced with high probability that a designated verifier proof intended to Bob is actually generated by Alice, as Bob would not or could not generate it himself. For example:

1. When Bob is believed to be honest, Cindy would trust that Bob does never deviate from his prescribed protocol, so that by seeing a signature, she would be convinced that it is originated by Alice.
2. When Cindy is sure that Bob has not yet seen a signature intended to himself, she would be convinced that the signature is not "forged" by Bob.

In these cases, we need a stronger notion of designated verifier proofs that is defined in [8] as follows:

Definition 3 (Strong Designated Verifier).
Let (P_A, P_B) be a protocol for Alice to prove the truth of the statement Ω to Bob. We say that Bob is a strong designated verifier if for any protocol (P_A, P_B, P_D, P_C) involving Alice, Bob, Dave and Cindy, by which Dave proves the truth of ϑ to Cindy, there is another protocol (P'_D, P_C) such that Dave can perform the 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).

Here again, all this definition amounts to saying is that the transcripts of a "real" proof may be simulated by anybody in such a way that they are indistinguishable for everybody other than Bob[1]. So, accordingly to our definition of designated verifier proofs, we define the strongness as follows:

Definition 4 (New Strong Designated Verifier).
Let $P(A, B)$ be a protocol for Alice to prove the truth of the statement Ω to Bob. We say that $P(A, B)$ is a strong designated verifier proof if anybody can produce identically distributed transcripts that are indistinguishable from those of $P(A, B)$ for everybody, except for Bob.

3 Related work

In this section, we first recall the designated verifier signature scheme [8] introduced by Jakobsson, Sako and Impagliazzo[2] (JSI, for short). Then we present a special case of the ring signature scheme [11] introduced by Rivest, Shamir and Tauman (RST, for short) that provides the designated verifier property[3].

Another scheme that may be turned into a designated verifier signature is due to Abe, Ohkubo and Suzuki [1] and is known as 1-out-of-n signature. Because of lack of space, we do not describe this scheme. We just notice that it is essentially some kind of ring signatures that may make use of different type of keys.

3.1 The JSI designated verifier scheme

The JSI scheme is a non-interactive designated verifier proof of Chaum's undeniable signatures. Let p be a large prime and q a prime divisor of $p-1$. Let g be a generator in $\mathbb{Z}_p^*$ of order q. A user u's private key will be denoted $x_u \in \mathbb{Z}_q^*$ and the corresponding public key $y_u = g^{x_u} \mod p$.

Signature generation. Alice wants to prove to Bob that $s = m^{x_a} \mod p$ is her signature on the message m. To prepare the poof, she selects w, r, t in $\mathbb{Z}_q$ and computes

$$\begin{aligned} &c = g^w y_b^r \mod p \\ &G = g^t \mod p \\ &M = m^t \mod p \\ &h = hash_q(c, G, M) \\ &d = t + x_a(h + w) \mod q \end{aligned}$$

where $hash_q$ denotes a hash function mapping values into $\mathbb{Z}_q$. Alice's signature on m and the associated proof is $\sigma = (s, w, r, G, M, d)$.

---

[1] Bob is actually the only party that can distinguish between real and simulated proofs. Even Alice cannot do so, if we assume that she does not keep the track of the proofs she generates.

[2] Chaum introduced a very similar scheme [3] under the name of private signatures.

[3] The main motivation of ring signatures is something more general than designing designated verifier signatures. The designated verifier property is rather a side effect of a particular setting of ring signatures.

Signature verification. Bob can verify that σ is a valid signature on the message m by computing

$$c = g^w y_b^r \mod p$$
$$h = hash_q(c, G, M)$$

and by checking whether

$$G y_a^{h+w} = g^d \mod p$$
$$M s^{h+w} = m^d \mod p.$$

If this holds, then Bob is convinced that Alice has generated this signature. However, other people (who can verify the consistency of the signature as Bob does) cannot conclude that Alice issued the signature, because they know that Bob can produce identically distributed transcripts, as follows.

Transcript simulation. To simulate transcripts of an Alice's signature, Bob chooses $d, \alpha, \beta \in \mathbb{Z}_q$ and computes

$$c = g^\alpha \mod p$$
$$G = g^d y_a^{-\beta} \mod p$$
$$M = m^d s^{-\beta} \mod p$$
$$h = hash_q(c, G, M)$$
$$w = \beta - h \mod q$$
$$r = (\alpha - w) x_b^{-1} \mod q.$$

In the appendix, we show how a third party, Cindy, can act as a middle person to prepare fake signatures under Alice's name intended to Bob. We also give a countermeasure to this problem.

3.2 Ring signatures

We only present here the restricted case of the ring signature of [11] that implements a designated verifier signature scheme. In [11], two versions are proposed. Here, we only consider the version based on the RSA public-key cryptosystem that is more efficient when the number of potential signers is fixed to two.

We denote by n_a, e_a and n_b, e_b Alice's and Bob's public keys, respectively, and by d_a and d_b the respective private keys.

Signature generation. When Alice wants to sign a message m for Bob, she chooses two random values $v, x_b \in \{0,1\}^c$, where 2^c is larger than both n_a and n_b. Then she computes $x_b^{e_b} \mod n_b$ and extends the result over $\{0,1\}^c$ (for a description of this extension, see [11], section 3.1. In the following, we assume that all the calculations are extended over $\{0,1\}^c$). Now, she solves the following equation in order to determine the value of y:

$$E_{h(m)}\left(x_b^{e_b} \mod n_b \oplus E_{h(m)}\left(y \oplus v\right)\right) = v$$

where $h(.)$ is a one-way hash function and $E_k(.)$ denotes a symmetric encryption with k as the key.

Using her private key, Alice goes on computing x_a, such that $x_a^{e_a} \mod n_a = y$. Alice's signature on message m is then $s = (v, x_a, x_b)$.

Signature verification. Anybody can verify the signature by simply checking whether

$$E_{h(m)}\left(x_b^{e_b} \mod n_b \oplus E_{h(m)}\left(x_a^{e_a} \mod n_a \oplus v\right)\right) = v.$$

Transcript simulation. The above signature will not convince Cindy that s is a valid Alice's signature. This is due to the fact that Bob can also compute a signature s', such that the verification succeeds. To do so, Bob chooses $v', x'_a \in \{0,1\}^c$, computes ${x'}_a^{e_a} \mod n_a$ and solves the equation

$$E_{h(m)}\left(y' \oplus E_{h(m)}\left({x'}_a^{e_a} \mod n_a \oplus v'\right)\right) = v'.$$

Now Bob computes x'_b, such that ${x'}_b^{e_b} \mod n_b = y$ and produces $s' = (v', x'_a, x'_b)$ which is undistinguishable from an Alice's signature.

Note that the public-keys e_a and e_b may be set to 3, which allows both exponentiations in the verification phase as well as one of the exponentiations in the signature generation to be replaced by two modular multiplications. So the scheme requires only one modular exponentiation altogether. This, however, is computed with respect to a large RSA exponent, that means that for an equivalent security level, an exponentiation in the RST scheme is more complex than 3 exponentiations in the JSI scheme (see section 7).

4 Description of the scheme

As is the case in all DL based schemes, we assume that some common parameters are initially shared between the users: a large prime p, a prime factor q of $p-1$, a generator $g \in \mathbb{Z}_p^*$ of order q and a one-way hash function h that outputs values in $\mathbb{Z}_q$.

Each user i chooses his secret key $x_i \in \mathbb{Z}_q$ and publishes the corresponding public key $y_i = g^{x_i} \mod p$.

Signature generation. In order to sign a message m for Bob, Alice selects two random values $k \in \mathbb{Z}_q$ and $t \in \mathbb{Z}_q^*$ and computes

$$\begin{aligned} c &= y_b^k \mod p, \\ r &= h(m, c), \\ s &= kt^{-1} - rx_a \mod q. \end{aligned}$$

The triple (r, s, t) is then the signature of the message m.

Verification. Knowing that a signature is originated from Alice, Bob may verify its validity by checking whether $h(m, (g^s y_a^r)^{tx_b} \mod p) = r$.

As we can see, nobody else other than Bob can perform this verification, since his secret key is involved in the verification equation. Hereafter, we show that even if Bob reveals his secret key, he cannot convince another party, Cindy, of the validity of such a signature.

Indeed when Cindy is given Bob's secret key, she can certainly check the consistency of the signature in the same way as Bob. But, there is no reason that she accepts it as an Alice's signature, because Bob is capable to generate the same transcripts in an indistinguishable way.

Transcript simulation. Bob selects $s' \in \mathbb{Z}_q$ and $r' \in \mathbb{Z}_q^*$ at random and compute

$$
\begin{aligned}
c &= g^{s'} y_a^{r'} \mod p \\
r &= h(m, c) \\
\ell &= r' r^{-1} \mod q \\
s &= s' \ell^{-1} \mod q \\
t &= \ell x_b^{-1} \mod q.
\end{aligned}
$$

Then $c = (g^s y_a^r)^{t x_b} \mod p$ and $h(m, c) = r$. In fact

$$
\begin{aligned}
(g^s y_a^r)^{t x_b} &\mod p = \\
(g^s y_a^r)^{\ell} &\mod p = \\
g^{s\ell} y_a^{r\ell} &\mod p = \\
g^{s'} y_a^{r'} &\mod p = c
\end{aligned}
$$

and $h(m, c) = r$ by definition.

Remarks:

1. To be complete, let us notice that Cindy should start by checking whether the received secret key is actually the Bob's one ($g^{x_b} \mod p \stackrel{?}{=} y_b$), because without it she is even not convinced that the signature is made by "Alice or Bob", as anybody may have simulated Alice's signature (intended to himself) and give his secret key to Cindy, claiming that it is the Bob's one.
2. Instead of revealing his secret key, Bob can prove to Cindy the consistency of a signature (and not that it is originated by Alice) as follows. Bob presents (m, s, t, c) to Cindy. Then Cindy computes $r = h(m, c)$ and asks to Bob to prove the knowledge of x_b as the discrete logarithm of y_b in one hand and of c on the other hand with respect to g and $(g^s y_a^r)^t \mod p$ as the bases, respectively.

5 Security

In [9] and [10], Pointcheval and Stern discussed the security of a large class of signature schemes, namely those that are derived from zero-knowledge identification protocols by replacing the verifier's role with some hash functions. They argued that in order to prove such schemes secure, it is essential to consider the hash function in use as a random function and showed how to prove the security using the Random Oracle Model [2] and a new lemma, called the Forking Lemma.

We assume that the reader is familiar with the random oracle model and the forking lemma. We just note that the security proofs in this model are based on the following statement: "If there exists a probabilistic polynomial time Turing machine that outputs with non-negligible probability a valid signature linked to a given user without knowing the related secret key, then a polynomial replay of that machine with the same random tape and a different oracle will produce two valid signatures, leading to the resolution of the hard problem on which the signature scheme relies".

At the first glance, our designated verifier signature scheme seems to satisfy all the necessary properties in order to be analysed in this model. However, this cannot be the case because of the designated verifier's capability of simulating signatures. Indeed in our scheme, Bob can simulate as many Alice's signatures as he desires (with himself as the designated verifier) without being able to derive Alice's secret key from them. What makes this impossible is that if Bob reuses the same random values r' and s', but with different oracles in simulating two signatures, he will only obtain linearly dependent transcripts that are clearly of no use for deriving Alice's secret key:

$$\begin{cases} c = g^{s'} y_a^{r'} \mod p \\ r = h(m, c) \\ \ell = r' r^{-1} \mod q \\ s = s' \ell^{-1} \mod q \\ t = \ell x_b^{-1} \mod q \end{cases} \qquad \text{and} \qquad \begin{cases} c = g^{s'} y_a^{r'} \mod p \\ \hat{r} = \hat{h}(m, c) \\ \hat{\ell} = r' \hat{r}^{-1} \mod q \\ \hat{s} = s' \hat{\ell}^{-1} \mod q \\ \hat{t} = \hat{\ell} x_b^{-1} \mod q \end{cases}$$

$$\begin{aligned} &\Longrightarrow (g^s y_a^r)^{t x_b} = (g^{\hat{s}} y_a^{\hat{r}})^{\hat{t} x_b} \\ &\Longrightarrow (g^{s'\ell^{-1}} y_a^{r'\ell^{-1}})^{\ell} = (g^{s'\hat{\ell}^{-1}} y_a^{r'\hat{\ell}^{-1}})^{\hat{\ell}} \\ &\Longrightarrow g^{s'} y_a^{r'} = g^{s'} y_a^{r'} \end{aligned}$$

Hence, since there is no assumption on the behaviour of the attackers in the random oracle model, some attackers may act as Bob and produce signatures such that a replay with the same random tape and a different oracle generates a second signature that is linearly dependent to the first one, i.e., without being able to derive Alice's secret key. This means that our scheme may not be proved secure in the above model.

Fortunately, the security proof in our case is even simpler than for classical signature schemes. In our model, we do not require the possibility of replaying the attacker or using two different oracles; all we need is the assumption that the hash function is considered as an oracle that on each valid input produces a random value and that this input should be known to the attacker beforehand. With this in mind, we prove that

Theorem 1. *If a valid Alice's signature for Bob (as the designated verifier) can be generated without the knowledge of Alice's or Bob's secret keys, then the Diffie-Hellman problem may be solved in polynomial time.*

Proof (sketch). Let T be a probabilistic polynomial time Turing machine that receives y_a, y_b and g as input. We assume that T may ask a polynomial number of questions to the random oracle and that they are of the form (m_i, c_i). We also assume that these questions are stored in a table together with the related answers.

Now suppose that T can find with non-negligible probability a valid signature (t, r, s) on a message m, such that r is the output of the oracle on the query (m, c), for some c that is computed or selected by T beforehand. We consider a variant T' of T that uses the generated signature to compute the Diffie-Hellman shared key of Alice and Bob, i.e., $g^{x_a x_b}$. To do so, T', after having seen the output of T, searches in the table of the stored questions and answers to find c corresponding to m and r. Now, T' can determine $g^{x_a x_b} = (c^{t^{-1}} y_b^{-s})^{r^{-1}}$, because

$$\begin{aligned}(c^{t^{-1}} y_b^{-s})^{r^{-1}} &= (((g^s y_a^r)^{t x_b})^{t^{-1}} y_b^{-s})^{r^{-1}} \\ &= ((g^s y_a^r)^{x_b} y_b^{-s})^{r^{-1}} \\ &= (y_b^s y_b^{r x_a} y_b^{-s})^{r^{-1}} \\ &= (y_b^{r x_a})^{r^{-1}} \\ &= y_b^{x_a} \\ &= g^{x_a x_b}.\end{aligned}$$

Remarks

3. The interesting question that arises from this proof is the following. If forging a single signature allows to determine $g^{x_a x_b}$, why the knowledge of a "real" signature does not lead to the calculation of this key. The answer is clear: with a real signature (t, r, s), nobody, without the knowledge of the designated verifier's secret key, can compute the related c that is required for the computation of the Diffie-Hellman shared key of two users. However, in order to forge a signature, one has, by assumption, to know this value in order to prepare the input of the oracle.
4. Once the value c related to a signature (t, r, s) on the message m becomes known to an attacker, it is possible for her to forge signatures under Alice's name for Bob on any message $\hat{m}$ of her choice. In fact, it suffices to compute $\hat{r} = h(\hat{m}, c)$ and set $\hat{s} = sd \mod q$ and $\hat{t} = td^{-1} \mod q$, where $d = \hat{r} r^{-1} \mod q$. In this case,

$$\begin{aligned}(g^{\hat{s}} y_a^{\hat{r}})^{\hat{t} x_b} &\mod p = \\ (g^{sd} y_a^{rd})^{td^{-1} x_b} &\mod p = \\ (g^s y_a^r)^{x_b} &\mod p = c\end{aligned}$$

and $\hat{r} = h(\hat{m}, c)$, by construction.
However, since this kind of signature is designed to be verifiable just for a particular verifier, it is only up to him to reveal c to other people and make them capable to fool him in the future.

Theorem 2. *The above signature scheme is designated verifier.*

Proof. We have to show that the transcripts simulated by Bob are indistinguishable from those that he receives from Alice, i.e., the following distributions are identical:

$$\sigma = (r, s, t) : \begin{cases} k \in_R \mathbb{Z}_q \\ t \in_R \mathbb{Z}_q^* \\ r = h(m, y_b^k \mod p), \\ s = kt^{-1} - rx_a \mod q \end{cases}$$

and

$$\sigma' = (r, s, t) : \begin{cases} s' \in_R \mathbb{Z}_q \\ r' \in_R \mathbb{Z}_q^* \\ r = h(m, y_a^{r'} g^{s'} \mod p), \\ s = s'r'^{-1}r \mod q \\ t = r'r^{-1}x_b^{-1} \mod q \end{cases}$$

Let $(\hat{r}, \hat{s}, \hat{t})$ be a signature that is randomly chosen in the set of all valid Alice's signatures intended to Bob. Then we have:

$$\Pr_{\sigma}[(r, s, t) = (\hat{r}, \hat{s}, \hat{t})] = \Pr_{k;t \neq 0} \begin{bmatrix} r = h(m, y_b^k \mod p) = \hat{r} \\ t = \hat{t} \\ s = kt^{-1} - rx_a \mod q = \hat{s} \end{bmatrix} = \frac{1}{q(q-1)}$$

and

$$\Pr_{\sigma'}[(r, s, t) = (\hat{r}, \hat{s}, \hat{t})] = \Pr_{s';r' \neq 0} \begin{bmatrix} r = h(m, y_a^{r'} g^{s'} \mod p) = \hat{r} \\ t = r'r^{-1}x_b^{-1} \mod q = \hat{t} \\ s = s'r'^{-1}r \mod q = \hat{s} \end{bmatrix} = \frac{1}{q(q-1)}$$

that means that both distributions of probabilities are the same.

6 Strong designated verifier signatures and practical issues

In [8], it is suggested that, in order to make designated verifier signatures strong, transcripts may be probabilistically encrypted using the public key of the intended verifier. This would guarantee that Cindy, who does not know the intended verifier's secret key, cannot verify such a signature, nor distinguish the transcripts from random strings of the same length and distribution.

This, however, requires the encryption of all or part of the transcripts with the intended verifier's public key, or alternatively the encryption of a session key with which the transcripts should be encrypted using a symmetric cipher, whence an additional complex operation.

As we noticed earlier, the verification of validity or invalidity of signatures in our scheme may only be performed by the designated verifier, since his secret key is involved in the verification. Therefore, all a third party may observe is a set of transcripts that are actually indistinguishable for her from random strings of the same length and distribution. This means that our scheme is inherently strong without any encryption or other operations.

In some circumstances, however, Alice may wish to encrypt the message itself, in order to prevent Cindy to read it (even though she knows that Cindy cannot get convinced of the origin of the message). With strong versions of the two schemes presented in section 3, if an encrypted session key is used for encrypting transcripts, the same session key may be used to encrypt the message. In other words, if the strongness is required, the privacy may be obtained at virtually no cost. In our scheme, since no encryption of transcripts is needed, a session key should still be established between Alice and Bob to encrypt and decrypt the message. But, this is already accomplished during the protocol: since both Alice and Bob can compute some value (namely c) without other parties knowing it, it may serve as a shared session key with which the message can be encrypted. In this case, instead of signing a message for Bob, Alice has to "signcrypt" it as follows.

She selects two random values $k \in \mathbb{Z}_q$ and $t \in \mathbb{Z}_q^*$ and computes $c = y_b^k \mod p$. Then she splits c into c_1 and c_2 of appropriate lengths and computes

- $z = E_{c_1}(m)$
- $r = h(m, c_2)$
- $s = kt^{-1} - rx_a \mod q$

where E is a symmetric encryption algorithm. (r, s, t, z) is then the signcryption of the message m.

Knowing that this signcrypted message is received from Alice, Bob

- computes $c = (g^s y_a^r)^{tx_b} \mod p$,
- splits c into c_1 and c_2 (as Alice does),
- finds $m = D_{k_1}(z)$, where D is the decryption algorithm, and
- verifies whether $h(m, c_2) = r$.

If so, Bob makes sure that m is the original message signcrypted by Alice.

7 Comparison

In this section, we give a performance comparison of our scheme and the two existing ones, namely the JSI and the RST schemes. For this comparison, we choose an implementation, setting $p = 512$ bits and $q = 160$ bits for the JSI and our scheme. In order to have comparable security, we set the RSA modulus to 512 bits in the RST scheme.

For the comparison to be effective, we only consider the number of modular exponentiations, which are the most time-consuming operations, and neglect other operations such as hashing, modular multiplications and symmetric encryptions. We also suppose that when using RSA, the public-key is set to 3, which allows one exponentiation to be replaced by two modular multiplications that are considered negligible.

	JSI	Strong JSI	RST	Strong RST	New scheme	Strong new scheme
Generation	1,200	+0	768	+0	240	+0
Verification	1,200	+768	0	+768	480	+0
Total	2,400	+768	768	+768	720	+0
Size (bits)	2,368	+512	1,536	+512	480	+0

Table 1. Performance and size comparison

In table 1, we indicate the complexity—in terms of modular multiplications resulting from modular exponentiations—of the two existing schemes in their strong and no strong flavours, as well as the new scheme. We assume that an exponentiation is equivalent to $1.5 \times \ell$ modular multiplications, where ℓ is the size of the exponent. In order for the JSI and the RST schemes to provide the strong designated verifier property, they need to be encrypted. We assume that a session key is encrypted using 512 bit RSA public-key encryption. This session key can then be used to cipher the transcripts.

We can see in table 1 that our scheme is much more efficient than the JSI scheme for both generation and verification. One may also see that the verification in the RST scheme is the most efficient. However this is not crucial for designated verifier signatures. In traditional signature schemes, efficient verification is a desirable design issue, motivated by the fact that a signature is generated once, but may be verified many times. In designated verifier schemes, there exists only one verifier, which implies only one verification. Therefore we argue that, for designated verifier schemes, only the total computational cost, regrouping signature generation and verification, is significant. When considering the total computing amount, our scheme is slightly more efficient than the RST scheme in the normal case, and more than twice more efficient in case of strong designated verifier.

Note that, our scheme may also be compared with the 1-out-of-n signature scheme [1]. If we consider, for instance, the discrete-logarithm case of this scheme (that uses the Schnorr signature scheme as the basis), our scheme is at least twice more efficient, even without requiring the strongness.

Finally, we would like to emphasize on the size of the respective signatures. Assuming that the hash function's output is of 160 bits, our scheme provides significantly smaller signatures compared with the two other, namely 3 times less than the RST signatures and 5 times less than those of the JSI.

8 Conclusion

In this paper, we proposed a new designated verifier signature scheme. To the best of our knowledge, it is the first scheme providing directly the strong designated verifier property, without any additional encryption. We introduced new definitions of designated verifier proofs and analysed the security of our scheme based on those definitions. We also compared the new scheme with the existing ones and showed that it is more efficient in terms of both computation and communication complexity, especially when the strongness is required.

References

1. M. Abe, M. Ohkubo and K. Suzuki, 1-out-of-n signatures from a variety of keys, Advances in Cryptology (Proceedings of Asiacrypt '02) , Lecture Notes in Computer Science, vol. 2501, Springer-Verlag, 2002, pp. 415-432.
2. M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, Proceedings of the 1st CCCS, ACM Press, 1993, pp. 62-73.
3. D. Chaum, Private signature and proof systems, United States Patent 5,493,614, 1996.
4. D. Chaum and H. Van Antwerpen, Undeniable signatures, Advances in Cryptology (Proccedings of Crypto '90), Lecture Notes in Computer Science, vol. 435, Springer-Verlag, 1991, pp. 212-216.
5. Y. Desmedt, C. Goutier and S. Bengio, Special uses and abuses of the Fiat-Shamir passport protocol, Advances in Cryptology (Proccedings of Crypto '87), Lecture Notes in Computer Science, vol. 293, Springer-Verlag, 1988, pp. 21-39.
6. Y. Desmedt and M. Yung, Weaknesses of Undeniable Signature Schemes (Extended Abstract), Advances in Cryptology (Proccedings of Eurocrypt '91), Lecture Notes in Computer Science, vol. 547, Springer-Verlag, 1992, pp. 205-220.
7. M. Jakobsson, Blackmailing using Undeniable Signatures, Advances in Cryptology (Proccedings of Eurocrypt '94), Lecture Notes in Computer Science, vol. 950, Springer-Verlag, 1995, pp. 425-427.
8. M. Jakobsson, K. Sako and R. Impagliazzo, Designated Verifier Proofs and Their Applications, Advances in Cryptology (Proceedings of Eurocrypt '96) , Lecture Notes in Computer Science, vol. 1070, Springer-Verlag, 1996, pp. 143-154.
9. D. Pointcheval and J. Stern, Security proofs for signature schemes, Advances in Cryptology (Proceedings of Eurocrypt '96), Lecture Notes in Computer Science, vol. 1070, Springer-Verlag, 1996, pp. 387-398.
10. D. Pointcheval and J. Stern, Security arguments for digital signatures and Blind signatures, Journal of Cryptology, vol. 13, no. 3, 2000, pp. 361-396.
11. R. Rivest, A. Shamir and Y. Tauman, How to Leak a Secret, Advances in Cryptology (Proceedings of Asiacrypt '01), Lecture Notes in Computer Science, vol. 2248, Springer-Verlag, 2001, pp. 552-565.
12. C. Schnorr, Efficient signature generation by smart cards, Journal of Cryptology, vol. 4, no. 3, 1991, pp. 161-174.
13. Y. Zheng, Digital signcryption or how to achieve cost(signature & encryption) $<<$ cost(signature) + cost(encryption), Advances in Cryptology (Proccedings of Crypto '97), Lecture Notes in Computer Science, vol. 1294, Springer-Verlag, 1997, pp. 165-179.

A Cryptanalysis and enhancement of the JSI scheme

The first requirement in a designated verifier signature scheme is that Bob, the intended verifier, should trust on the authenticity and the origin of the signatures he receives. Hereafter, we show that when Alice signs a message m for Bob as the intended recipient, Cindy can transform it to an Alice's signature on a message m' of her choice without Bob being able to detect the forgery.

More precisely, Cindy observes a potential "message,signature" pair (m, s) with its associated proof $\sigma = (w, r, G, M, d)$, and replaces (m, s) with a "chosen-message,fake signature" pair (m', s'), such that the proof is left intact. To do so, Cindy computes $s' = (m'^d M^{-1})^{(h+w)^{-1}} \mod p$. Then Bob, who recieves the pair (m', s') and the proof σ, will obviously accept it as originated from Alice.

The reason that this attack works is because neither m' nor s' are connected in any way to the transcripts of σ, i.e., m' can be freely chosen and s' can be computed accordingly without affecting any part of the proof. So, to overcome this shortcoming, one of these values should be bound to σ, by means of the hash function. That is, if h is computed as $hash_q(c, G, M, s)$, then the new value of s (s' corresponding to the chosen m') would modify h, that is needed to be fixed in advance for computing s'.

Note that without putting m in the hash function, one may still fix s', compute h and derive the corresponding m' as $m' = (Ms'^{h+w})^{d^{-1}} \mod p$, i.e., the scheme would still be subject to existential forgery. However, since the JSI scheme is a non-interactive designated verifier proof of Chaum's undeniable signatures, it is assumed that m is already the result of a hashing, i.e., m =hash(original message). Therefore, once m' is computed as above, the attacker is left with a hard problem to derive the original messages, i.e., inversing the hash function.

Note also that this attack is not effective with the strong version of the JSI scheme assuming that the transcripts are encrypted.