Server-Aided Verification: Theory and Practice

Marc Girault and David Lefranc

France Telecom
Research and Development
42 rue des Coutures
F-14066 Caen, France
{marc.girault,david.lefranc}@francetelecom.com

Abstract. We introduce the server-aided verification (SAV) concept, which consists in speeding up the verification step of an authentication/signature scheme, by delegating a substantial part of computations to a powerful (but possibly untrusted) server. After giving some motivations for designing SAV protocols, we provide a simple but realistic model, which captures most situations one can meet in practice (note that this model is much more general than the one recently proposed by Hohenberger and Lysyanskaya, who require the server to be made of two softwares which do not communicate with each other [13]). Then, we analyze and prove in this model the security of two existing SAV protocols, namely the Lim-Lee [14] modification of Schnorr scheme [25] and the Girault-Quisquater variant [9] of GPS scheme [6, 22]. Finally, we propose a generic method for designing SAV versions of schemes based on bilinear maps, which can be applied to the Boneh-Boyen signature schemes [3], the Zhang-Safavi-Naini-Susilo [29] signature scheme and the Shao-Lu-Cao identification scheme [27].

Key words: identification protocol, digital signature, interactive proof, zero-knowledge, discrete logarithm, non-repudiation, bilinear map, pairing.

1 Introduction

Designing efficient signature and identification (or authentication) schemes is one of the main challenges of public key cryptography. Indeed, with the development of smart cards and RFID tags, or more generally low cost chips with small computation capabilities, it becomes crucial to propose schemes suited to such devices.

Until now, research has essentially focused on speeding up the prover/signer computations. In particular, zero-knowledge (ZK) paradigm [11] has led to very efficient schemes, including Fiat-Shamir [5], Guillou-Quisquater [12], Schnorr [25] and GPS [6, 22] protocols. Another research direction was to speed-up RSA [24] private operation by sharing the computation task with a powerful but untrusted computer [15]. The latter approach, usually named "server-aided RSA", is delicate and many proposals have been broken (see e.g. [18–20]).

On the other hand, speeding up the verification task has been relatively few scrutinized. One reason may be that RSA is very efficient from this point of view. Indeed, the use of a small public exponent leads to a few modular multiplications to perform, and even only one in the variant from Rabin [23]. Furthermore, the task of doing a modular reduction can be reduced to a still lighter operation by using various tricks (e.g. [28]). Finally, batch verification allows a server to verify many RSA signatures in an optimized manner ([1]).

Nevertheless, RSA leaves open the problem of designing an identification scheme which is efficient at both ends. A first valuable solution may be ZK factorization-based schemes, as they generally involve a moderate and well-balanced computational task at each end. However, in many environments, this task still remains too heavy. For instance, if the prover's device is a low cost smart card and/or if the transaction must take place in very few milliseconds, then significantly faster computations are required.

Discrete-logarithm-based schemes, when used in their "coupon mode" ([17]), allow the signer to compute the signature in a particularly fast manner. A coupon is essentially a precomputed exponential, leaving the signer with an ultra-light task (typically a modular multiplication with a small modulus or even less) at the time of signing. The verifier has a price to pay, since he must compute the exponential of a rather large number. As a consequence, speeding up verification step in a discrete-logarithm-based scheme is a very significant challenge, both from theoretical and practical points of view.

In a client-server environment, this challenge can be solved by using precomputation techniques, which allow to decrease the average time of verification by a factor close to five [4]. A more general approach would consist in sharing the computations with a powerful but untrusted server. Since it is the analogue of "server-aided RSA" at the verifier's side, we call it "server-aided verification". The main difference between these two notions is that the verification does not involve private keys, so that there is no concern with *confidentiality*. On the other hand, there is a major concern with *integrity*: the output of verification step must be trusted by the verifier -or it has no value at all.

In real world there are many situations in which a secure but not powerful chip is connected to a powerful but not secure powerful device. For example, in a GSM mobile telephone, the more sensitive cryptographic operations are performed in the so-called SIM (Subscriber Identification Module), which is already aided by the handset chip, mainly to decipher the over-the-air enciphered conversation. In a payment transaction, a so-called SAM (Secure Access Module) is embedded in a terminal already containing a more powerful chip. We can also mention the example of a smart card plugged into a personal computer, seeing that many PCs will be equipped with smart card readers in a near future.

This paper is about server-aided verification in general, applied to discrete-logarithm-based schemes in particular. While not using this name, nor providing formal definitions and proofs, a few authors have already addressed this problem. In the late 80's, at Cardis 2000 conference, Quisquater and De Soete proposed tricks to speed up RSA verification with a small exponent [28]. At Eurocrypt'95 conference, Lim and Lee gave a generic method, based on the "randomization" of the verification equation [14]. This equation is only known to the verifier, so that a cheater can solve it only if he guess it: security is unconditional. The counterpart of this nice method is that the verifier must perform a heavy precomputation before carrying out the transaction.

A completely different approach has been proposed by Girault and Quisquater [9] and described in European NESSIE project final proceedings [8]. While no precomputation nor randomization is required, the security remains computational, based on the hardness of a sub-problem (namely factorization) of the initial underlying problem (namely composite discrete logarithm).

Recently, at TCC'05, Hohenberger and Lysyanskaya addressed the situation in which the server is made of two untrusted softwares, which are assumed not to communicate with each other [13]. While this assumption is very particular and much stronger than ours (since, in essence, we assume that the verifier does not trust anybody nor anything except himself), it allows a very light public computation task (typically one modular multiplication in the Schnorr scheme).

Starting from this background, the motivation of this paper was two-fold: first, clarify the properties that server-aided verification must satisfy and classify the different approaches to achieve them; second, extend the class of protocols for which verification can be server-aided. The result is as follows: in section 2 we define our model and identify three possible approaches to achieve the main property. In section 3 we prove in this model the security of the two state-of-the art protocols mentioned above. In section 4 we propose a generic method applicable to signatures schemes based on bilinear maps (in particular Boneh-Boyen [3] and Zhang-Safavi-Naini-Susilo [29]), and prove them secure in our model. Finally, we provide a conclusion.

2 Model

2.1 An Illustrative Example

Let us illustrate the notion of server-aided verification with RSA signature scheme. In this scheme, the signer computes a signature σ of the message m by extracting an e^{th} root modulo n of $f(m)$, where f is specific to the exact scheme which is used (typically f is a combination of some hash-function(s) and some redundancy function(s)). The verifier checks that $\sigma^e \bmod n = f(m)$. If the equality holds, σ is accepted; otherwise, it is rejected.

If the verifier has only a small computation capability, he may want to be aided when checking the equality. So let us suppose that he has access to a more powerful, but untrusted server or, equivalently, to a trusted server via a non authenticated communication link. (If the server and the communication link were both trusted, then the verifier would just ask the server for the verification of $\sigma^e \bmod n = f(m)$ and the server would reply with OK or NOK). For instance, the verifier may think of the two following possibilities:

– Ask the server to compute the value $Y = \sigma^{\frac{e-1}{2}} \bmod n$. Then, the server returns the value Y and the verifier finally checks if $f(m) = Y^2\sigma \bmod n$.
– If the value $e = e_1 \times e_2$, with $e_1 < e_2$, then ask the server for the computation of $Y = \sigma^{e_2} \bmod n$ so that the verifier only has to check if $f(m) = Y^{e_1} \bmod n$.

For each of these protocols, even if it deviates from its assigned part, the server cannot fool the verifier as long as it does not collude with the (legitimate or not) prover. Indeed, given n, m and σ, finding Y such that the final verification equation is satisfied requires the ability to extract (square or e_1^{th}) roots mod n.

Now, what about a possible collusion between a cheating prover and the server? In the first protocol, an illegitimate prover and the server can easily collaborate in order to let the verifier accept a fake signature: the cheater can choose randomly $0 < \tilde{Y} < n$ and computes $\tilde{\sigma} = f(m)\tilde{Y}^{-2} \bmod n$. The cheater first provides the value $\tilde{\sigma}$, and then asks the server to answer with the value $\tilde{Y}$. Obviously, $\tilde{\sigma}$ and $\tilde{Y}$ fulfill $\tilde{Y}^2\tilde{\sigma} = f(m) \bmod n$ and, as a consequence, the signature is accepted. Since the solution to be chosen must at least resist a collusion between the server and an illegitimate prover (which in practice, may be represented by the same entity), this implies that this first protocol cannot be used.

Fortunately, this attack does not apply to the second protocol. Indeed, it remains secure against a coalition between a cheating prover and the server since finding the right value of Y given n and m requires to be able to extract an e_1^{th} root modulo n.

Nevertheless, a more subtle attack remains possible if the legitimate prover and the server collaborate. Instead of providing the verifier with the right signature σ, the legitimate prover may send a value $\tilde{\sigma}$, correlated to σ in a way known by the server (e.g. by adding it to a random value). Later, when the verifier wants to check the signature, he sends a request to the server. The latter reconstructs σ from n, m and $\tilde{\sigma}$, which allows him to reply with the right value Y. The verifier is satisfied and stores the pair $(m,\tilde{\sigma})$. Unfortunately, the stored signature is wrong and the legitimate prover can easily repudiate it. Here, the point is that the verifier does not check in any way the consistency between $\tilde{\sigma}$ and Y.

This leads us to distinguish, in the model that we propose, two kinds of deviating provers: on one hand the cheaters (who do not know the signature private key); on the other hand the legitimate provers (who misbehave in order to make some kind of repudiation possible).

Note however that the latter attack may be of very little significance in the case of an authentication protocol or more generally when non-repudiation property is not required.

2.2 Definitions

In this subsection, we formalize the server-aided verification concept. We deliberately restrict our model to the situation in which the modification of the basic protocol does not impact prover's view of the initial protocol. Indeed, the prover should not matter whether verification is server-aided or not. Of course, such an assumption does not prevent the prover from playing himself the role of the server if both parties find convenient to do so.

We also assume for simplicity that verification in the initial protocol consists in checking a predicate at the end of the protocol. Most of the protocols we are aware work that way, or can be replaced by equivalent protocols which work that way. First, we precisely define the different types of provers we consider.

Definition 1 (Legitimate/Misbehaving/Cheating). *Let π be an interactive proof of knowledge between a (so-called) prover $\mathcal{P}$ and a (so-called) verifier $\mathcal{V}$. As usual, we denote by $\tilde{\mathcal{P}}$ a prover which deviates from the protocol. The (non-deviating) prover $\mathcal{P}$ is said legitimate, while $\tilde{\mathcal{P}}$ is:*

cheating *if he does not hold the knowledge he claims to hold (and we denote him $\tilde{\mathcal{P}}_1$);*
misbehaving *if he holds this knowledge (and we denote him $\tilde{\mathcal{P}}_2$).*

We model the prover $\mathcal{P}$, the verifier $\mathcal{V}$ and the server $\mathcal{S}$ with polynomial probabilistic Turing machines (PPTM) with respective random tapes ω_P, ω_V and ω_S. We also consider that the verifier owns an additional tape denoted ω'_V on which precomputed values can be written. The *computational cost* of the verifier is defined as the number of steps performed by the Turing machine $\mathcal{V}$. In practical examples, our computation unit is the n-bit modular multiplication, which is a $\mathcal{O}(n^2)$ operation if using a standard algorithm to make it.

We now give the definition of a SAV protocol taking into account an optional property.

Definition 2 (SAV protocol). *Let π be an interactive proof of knowledge between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, with a common input $\mathcal{I}$ of size $|\mathcal{I}|$, and which halts by verifying a predicate $\mathcal{E}_\pi$ (we denote $\mathcal{E}_\pi = 1$ if the predicate is satisfied, $\mathcal{E}_\pi = 0$ if not). Let π-cost denote the computational cost of $\mathcal{V}$ during the execution of π.*

Let π^* be an interactive protocol between the prover $\mathcal{P}$, the verifier $\mathcal{V}$ and a server $\mathcal{S}$, equal to the composition of two protocols π^- and π' such that:

- *the protocol π^- is equal to the protocol π without the verification of $\mathcal{E}_\pi$;*

- *the protocol π' is an interactive protocol between $\mathcal{V}$ and $\mathcal{S}$;*
- *$\mathcal{V}$ finally accepts (π^*-accepts) or rejects (π^*-rejects) $\mathcal{I}$ by verifying a final predicate $\mathcal{E}_{\pi^*}$ (we denote $\mathcal{E}_{\pi^*} = 1$ if the predicate is satisfied, $\mathcal{E}_{\pi^*} = 0$ if not).*

Let π^*-cost denote the computational cost of $\mathcal{V}$ during the execution of π^*.
The protocol π^* is said to be a ***server-aided verification (SAV) protocol for*** *π if:*

1. *(auxiliary completeness)*
   - $\forall \mathcal{I}, \underset{\omega_{\mathcal{S}},\omega_{\mathcal{V}}}{Pr}(\mathcal{V}\ \pi^*\text{-accepts } \mathcal{I} \mid \mathcal{E}_\pi = 0)$ *is negligible in* $|\mathcal{I}|$.
   - $\forall \mathcal{I}, \underset{\omega_{\mathcal{S}},\omega_{\mathcal{V}}}{Pr}(\mathcal{V}\ \pi^*\text{-rejects } \mathcal{I} \mid \mathcal{E}_\pi = 1)$ *is negligible in* $|\mathcal{I}|$.
2. *(auxiliary soundness)* $\forall \mathcal{I}, \forall \tilde{\mathcal{P}}_1, \forall \tilde{\mathcal{S}}, \underset{\omega_{\tilde{\mathcal{P}}_1},\omega_{\mathcal{V}},\omega_{\tilde{\mathcal{S}}}}{Pr}(\mathcal{V}\ \pi^*\text{-accepts } \mathcal{I} \mid \mathcal{E}_\pi = 0)$ *is negligible in* $|\mathcal{I}|$.
3. *(computational gain) The computational cost π^*-cost is strictly less than π-cost.*

If non-repudiation is required, π^* must also verify:

(auxiliary non-repudiation) $\forall \mathcal{I}, \forall \tilde{\mathcal{P}}_2, \forall \tilde{\mathcal{S}}, \underset{\omega_{\tilde{\mathcal{P}}_1},\omega_{\mathcal{V}},\omega_{\tilde{\mathcal{S}}}}{Pr}(\mathcal{V}\ \pi^*\text{-accepts } \mathcal{I} \mid \mathcal{E}_\pi = 0)$ *is negligible in* $|\mathcal{I}|$.

Remark 1. From the definition, proving the auxiliary non-repudiation also proves the auxiliary soundness which also proves the first condition of the auxiliary completeness.

2.3 Achieving the auxiliary soundness

We currently distinguish two ways to achieve the auxiliary soundness.

Final predicate hard to know. The final predicate $\mathcal{E}_{\pi^*}$ is constructed from the predicate $\mathcal{E}_\pi$ by (secretly) randomizing it, so that only the verifier knows it. As a consequence, if the prover is a cheater $\tilde{\mathcal{P}}_1$ (and the predicate $\mathcal{E}_\pi$ is false), there is no way for the server $\tilde{\mathcal{S}}$ to fool the verifier, even if colluding with $\tilde{\mathcal{P}}_1$. The better strategy consists in guessing the randomizing parameter(s), the probability of which can be easily controlled by choosing the adequate size. Moreover, this case can be subdivided into two sub-cases:

- *Unconditionally* unknown predicate: even with unlimited resources, an enemy has no better strategy to retrieve the final predicate than guessing it.
- *Computationally* unknown predicate: it is computationally hard for the enemy to retrieve the final predicate.

Final predicate hard to solve. The final predicate $\mathcal{E}_{\pi^*}$ is constructed from $\mathcal{E}_\pi$ such that the final predicate is computationally hard to solve. As a consequence, if the prover is a cheater $\tilde{\mathcal{P}}_1$ (and the predicate $\mathcal{E}_\pi$ is false), there is no feasible way for the server $\tilde{\mathcal{S}}$ to fool the verifier, even if colluding with $\tilde{\mathcal{P}}_1$. Note that the hard problem used in $\mathcal{E}_{\pi^*}$ is necessarily easier than (i.e. can be reduced to) the one used in $\mathcal{E}_\pi$, since auxiliary completeness implies that a solution for the initial predicate can be turned into a solution for the final one.

2.4 Achieving the auxiliary non-repudiation.

A SAV protocol based on an unconditionally unknown predicate verifies the auxiliary non-repudiation since the security is unconditional (even w.r.t. $\tilde{\mathcal{P}}_2$).

In the case of a SAV protocol π^* based on a hard-to-solve predicate, if the hard-computational problem used for π^* can be broken using the private key involved in the basic protocol π, then a misbehaving prover can obviously break the SAV protocol so that the auxiliary non-repudiation is not verified.

2.5 Security model in the case of signature scheme

To prove that the auxiliary soundness (respectively the auxiliary non-repudiation) of a protocol π^*, we assume that $\tilde{\mathcal{S}}$ communicates with $\tilde{\mathcal{P}}_1$ (respectively $\tilde{\mathcal{P}}_2$). In the case of signature schemes, to make a message m and an invalid signature $\tilde{\sigma}$ be π^*-accepted, as in classical proofs [10] of signature scheme, we distinguish different type of attacks. Thus, we assume that $\tilde{\mathcal{P}}_1$ (respectively $\tilde{\mathcal{P}}_2$):

(No message attack) has no access to valid signatures.
(Known message attack) obtains a set of valid signatures associated to a given set of messages.
(Generic message attack) obtains a set of valid signatures associated to a set of messages of his choice, before knowing the public key.
(Directed chosen message attack) obtains the set of valid signatures associated to a set of messages of his choice, before knowing the public key.
(Adaptive chosen message attack) obtains valid signatures for messages of his choice, after knowing the public key.

In all these attacks, the message m, on which $\tilde{\mathcal{P}}$ sends an invalid signature $\tilde{\sigma}$, does not belong to the set of messages for which a valid signature is known.

3 SAV Protocols for Identification Schemes

In this section we apply our model to two formerly suggested methods to delegate computations during the verification step. The first one is a generic method proposed by Lim and Lee [14] and applicable to many discrete-logarithm-based schemes. The second one is a modification by Girault and Quisquater [9] of the GPS scheme, which is particularly interesting in the so-called RSA-like version of GPS, as proposed by Girault and Paillès [7]. We prove that these two solutions are both SAV protocols in the sense of section 2.2.

3.1 An Unconditionally-Unknown-Predicate-Based SAV Protocol

At Eurocrypt'95, Lim and Lee suggested a general method to delegate computations during the verification step of discrete-logarithm-based schemes (signature and identification) [14]. In this paper, the computations are not delegated to any server, but to the prover himself. Our approach is more general but their results can be easily adapted to it. To validate (or not) their generic method, we analyze it when applied to the Schnorr identification scheme; the results presented below remain valid for the Schnorr signature scheme.

The Lim-Lee protocol. In the Schnorr identification scheme, a prover owns a private key s and a public key (g, p, q, v) such that g is of prime order q modulo a prime integer p and $v = g^{-s} \bmod p$ (generally, g, p and q are system parameters). The Lim-Lee protocol is described in Fig. 1. In particular, it requires the precomputations of two values Z and $k^{-1} \bmod q$.

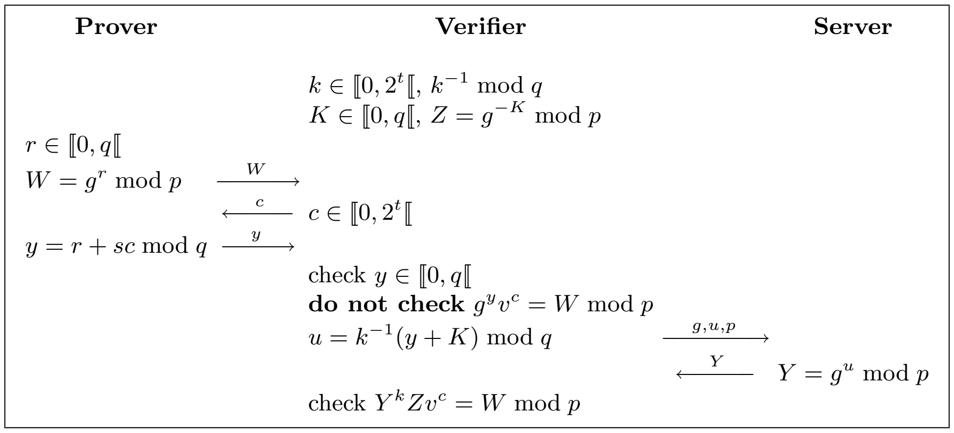
Prover
Verifier
Server
$k \in [\![0, 2^t[\![$, $k^{-1} \bmod q$
$K \in [\![0, q[\![$, $Z = g^{-K} \bmod p$
$r \in [\![0, q[\![$
$W = g^r \bmod p$ $\xrightarrow{W}$
$\xleftarrow{c}$ $c \in [\![0, 2^t[\![$
$y = r + sc \bmod q$ $\xrightarrow{y}$
check $y \in [\![0, q[\![$
do not check $g^y v^c = W \bmod p$
$u = k^{-1}(y + K) \bmod q$ $\xrightarrow{g,u,p}$
$\xleftarrow{Y}$ $Y = g^u \bmod p$
check $Y^k Z v^c = W \bmod p$

Fig. 1. The Lim-Lee modification of the Schnorr identification scheme

Theorem 1. *Let $\mathcal{I}$ be a public key (g, p, q, v) and t the security parameter for the Schnorr scheme. The Lim-Lee protocol is a SAV protocol for the Schnorr Scheme if $|q| > t$ and $\log_2 |\mathcal{I}| = o(t)$.*

Proof. Let $\mathcal{E}_\pi$ and $\mathcal{E}_{\pi^*}$ denote respectively the verification of $\big(y \in [\![0, q[\![$ and $g^y v^c = W \bmod p\big)$ and $\big(y \in [\![0, q[\![$ and $Y^k Z v^c = W \bmod p\big)$.
Auxiliary completeness. In the Lim-Lee method, $(\mathcal{E}_\pi = 1) \Longleftrightarrow (\mathcal{E}_{\pi^*} = 1)$ stands so that $\underset{\omega_\mathcal{S},\omega_\mathcal{V}}{Pr}$ ($\mathcal{V}$ π^*-accepts $\mathcal{I} \mid \mathcal{E}_\pi = 0$)$= 0$ and $\underset{\omega_\mathcal{S},\omega_\mathcal{V}}{Pr}$ ($\mathcal{V}$ π^*-rejects $\mathcal{I} \mid \mathcal{E}_\pi = 1$)$= 0$.

Auxiliary soundness. We assume $\mathcal{E}_\pi = 0$. From the given values u and y such that $u = k^{-1}(y + K) \bmod q$, with $k \in [\![0, 2^t[\![$ and $K \in [\![0, q[\![$, the entropy over k, in the sense of the Shannon theory, is then exactly equal to t so that k is unconditionally unknown. Since only one value k satisfies the final equation $\mathcal{E}_{\pi^*}$, the probability that $\mathcal{V}$ π^*-accepts is equal to to 2^{-t} (the probability of guessing the right k). This probability is negligible if $\log_2 |\mathcal{I}| = o(t)$.

Computational gain. In section 2.2, we assumed that the Turing machine associated to the verifier owns an additional tape to store precomputed values so that, when considering the computational cost of the verifier, these precomputed values are not taken into account.

Remark 2. As described in [25], by using an extension of the square and multiply algorithm with the precomputed value gv, the computation of $g^y v^c \bmod p$, with $|y| = l$, $|c| = t$ requires on average $1.5l + 0.25t$ modular multiplications.

In the Schnorr scheme, since $|y| = |q|$ and $|c| = |k| = t$, π-cost is equal to $1.5|q| + 0.25t$ modular multiplications. In the Lim-Lee protocol, computing $Y^k v^c$ requires $1.75t$ modular multiplications and multiplying by Z requires one more, i.e π^*-cost is equal to $1.75t + 1$ modular multiplications. If we omit the negligible cost of the modular multiplication $k^{-1}(y + K) \bmod q$, the difference of computational costs, given by $1.5(|q| - t) - 1$, is greater than 0 if $|q| > t$.

Auxiliary non-repudiation. As the security of the SAV relies on the perfect privacy of k, i.e the unconditional security of the transformation over y, even the misbehaving prover has no advantage over a cheater to determine this value k. $\square$

Example 1. Typically, q is a 160-bit number and t is equal to 32. Thus, the Lim-Lee protocol decreases by around 75% the computational cost of the verifier.

Remark 3. If, for one reason or another, the size of q is significantly larger than 160 bits, we can also design a SAV protocol based on a computationally unknown predicate. The verifier precomputes $h = g^K \bmod p$ with K typically a 160-bit number; the server computes $Y = h^y \bmod p$ and the verifier finally checks if $Y = (Wv^{-c})^K \bmod p$. This SAV protocol does not verify the auxiliary non-repudiation.

3.2 A Hard-to-Solve-Predicate-Based SAV Protocol

The protocol. The RSA-like GPS identification scheme was presented by Girault and Paillès at WCC'03 [7]. It is a variant of the GPS identification scheme [6, 22], based on the intractability of the RSA problem. In this scheme, a prover $\mathcal{P}$ owns a public key (n,g,f,e) and a private key s such that e is a prime integer and $es = 1 \bmod \phi(n)$, $f = g^{-s} \bmod n$ and, as a consequence, $g = f^{-e} \bmod n$.

In the same paper [7], Girault and Paillès suggest a method to delegate to a server some computations of the verification step. This protocol, called *server-aided RSA-like GPS identification scheme*, is described in Fig. 2. This method is essentially the same as the one proposed by Girault and Quisquater at Eurocrypt 2002 rump session and applied to the basic GPS identification scheme; it can be found in [8].

$$\begin{array}{lclcl}
\textbf{Prover} & & \textbf{Verifier} & & \textbf{Server} \\
r \in \llbracket 0, 2^R \llbracket & & & & \\
W = g^r \bmod n & \xrightarrow{W} & & & \\
 & \xleftarrow{c} & c \in \llbracket 0, e \llbracket & & \\
y = r + sc & \xrightarrow{y} & & & \\
 & & \text{Check } y \in \llbracket 0, 2^R + en \llbracket & & \\
 & & \textbf{do not check } g^y f^c = W \bmod n & \xrightarrow{f,y,n} & \\
 & & & \xleftarrow{Y} & Y = f^{-y} \bmod n \\
 & & \text{check } Y \in \rrbracket 0, n \llbracket & & \\
 & & \text{check } Y^e f^c = W \bmod n & &
\end{array}$$

Fig. 2. The server-aided RSA-like GPS identification scheme

In the following, we denote π the RSA-like GPS identification scheme and π^* the server-aided RSA-like GPS identification scheme.

Theorem 2. *Let $\mathcal{I}$ be a public key (n,g,f,e) for π. Under the intractability of the RSA problem, π^* is a SAV protocol for π if $\log_2|\mathcal{I}| = o(\log_2 e)$.*

Proof. We denote $\mathcal{E}_\pi$ and $\mathcal{E}_{\pi^*}$ the respective verification of equations of $\big(y \in [\![0, 2^R + en[\![$ and $g^y f^c = W \bmod n\big)$ and $\big(y \in [\![0, 2^R + en[\![$ and $Y \in]\!]0, n[\![$ and $Y^e f^c = W \bmod n\big)$.
Auxiliary completeness. $(\mathcal{E}_\pi = 1) \iff (\mathcal{E}_{\pi^*} = 1)$ stands so that, using the same argument as in the proof of the Lim-Lee protocol, the auxiliary completeness is verified.

Auxiliary soundness. We assume $\mathcal{E}_\pi = 0$. The proof relies on the following lemma.

Lemma 1. *Assuming $\mathcal{E}_\pi = 0$, if $\mathcal{I}$ is π^*-accepted with a probability greater than $\varepsilon' = \frac{1}{e} + \varepsilon$, then, an e^{th} root of the public key f can be computed in time less than $4\tau/\varepsilon'$, with a probability greater than $\frac{\varepsilon^2}{6(\varepsilon')^2}$, τ denoting the average running time of π^*.*

Proof. Assuming $\mathcal{E}_\pi = 0$, with a classical consideration (see e.g [12]), the probability of success in π^* is at least $1/e$. This probability is negligible in $|\mathcal{I}|$ if $\log_2|\mathcal{I}| = o(\log_2 e)$. Let us assume that

$$\Pr_{\omega_{\tilde{\mathcal{P}}_1}, \omega_{\mathcal{V}}, \omega_{\tilde{\mathcal{S}}}} (\mathcal{V}\ \pi^*\text{-accepts } \mathcal{I} \mid \mathcal{E}_\pi = 0) \geq 1/e + \varepsilon = \varepsilon'.$$

In appendix A, we describe an algorithm which returns, in time less than $4\tau/\varepsilon'$ and with a probability greater than $\frac{\varepsilon^2}{6\varepsilon'^2}$, two triplets (W,c_1,Y_1) and (W,c_2,Y_2) with $c_1 \neq c_2$, $Y_1 \neq 0$ and $Y_2 \neq 0$.

These two triplets verify $Y_1^e f^{c_1} = Y_2^e f^{c_2} \bmod n$. Without loss of generality, we assume that $c_2 > c_1$ so that $0 < c_2 - c_1 < e$. Since e is a prime integer, $gcd(e, c_2 - c_1) = 1$, so that there exist two integers a and b such that $a \times e + b \times (c_2 - c_1) = 1$. Then, we obtain $f = \big(f^a (Y_1/Y_2)^b\big)^e \bmod n$ so that $f^a(Y_1/Y_2)^b \bmod n$ is an e^{th} root of the public key f. $\square$

Computational gain. Let l denote the binary size of y; using Remark 2, the computation cost of the verifier in π^* is equal to $1.75|e|$ whereas in π, it is equal to $1.5l + 0.25|e|$. Due to the absence of modular reduction in y, the difference of computational costs, given by $1.5(l - |e|)$, is greater than 0.

Auxiliary non-repudiation. A misbehaving prover $\tilde{\mathcal{P}}_2$ involved in π^* knows the inverse s of e modulo $\phi(n)$. Thus, for this prover, solving the RSA problem is easy, and the auxiliary non-repudiation cannot be verified. $\square$

Example 2. Typically, n is a 1024-bit number, e is a 32-bit number and s is a 1024-bit number. No modular reduction is required so that y is around a 1050-bit number. Using the server-aided RSA-like GPS scheme, the computational cost of the verifier is decreased by around 95%.

Remark 4. Two other SAV protocols for GPS family can be considered; the first one verifies the auxiliary non-repudiation and is in the Lim-Lee article [14]; the second one is the same as in Remark 3.

4 First SAV Protocols for Pairing-Based Schemes

In this part, we present a generic SAV protocol applicable to several signature schemes based on bilinear maps (or pairings): in particular, the ZSNS signature scheme from Zhang, Safavi-Naini and Susilo [29] and the Boneh-Boyen signature schemes [3] (it also applies on the identification scheme from Shao, Lu and Cao [27] which is constructed from one of the Boneh-Boyen signature schemes).

4.1 The Generic SAV Protocol

We assume the existence of a bilinear map $e : G \times G \to G_1$. The groups G and G_1 are cyclic groups of prime order p and $e(g, g) \neq 1$.

The method applies on schemes in which the verifier checks if $e\big(\sigma, f(\mathcal{I}, m, r)\big) = e(g, g)$, such that f is a public function specific to the scheme, $\mathcal{I}$ the public parameters including the public key, $(r,sigma)$ defines the signature of a message. Depending of the scheme, r exists or not.

The basic idea of our generic SAV protocol is to delegate the (expensive) pairing execution to the server, the verifier being left with two (relatively much faster) exponentiations, one of them being precomputable. On the whole, it consists in first executing the signature scheme and, instead of verifying the equation $e\big(\sigma, f(\mathcal{I}, m, r)\big) = e(g, g)$, in running the following protocol with a server.

1. The verifier randomly picks an integer $t \in [\![0, p[\![$, and precomputes $e(g, g)^t$.
2. After receiving a message and its signature, the verifier computes the value $\alpha = \big(f(\mathcal{I}, m, r)\big)^t$ and sends it with σ to the server.
3. The server computes the value $\beta = e(\sigma, \alpha)$ and sends it back to the verifier.
4. The verifier finally checks if $\beta = e(g, g)^t$.

Auxiliary completeness. The equivalence $e\big(\sigma, f(\mathcal{I}, m, r)\big) = e(g, g) \Longleftrightarrow e(\sigma, \alpha) = e(g, g)^t$ stands and using the same argument as in the proof of the Lim-Lee protocol, the auxiliary completeness is verified.

Auxiliary non-repudiation. Our SAV construction allows the misbehaving prover $\tilde{\mathcal{P}}_2$ to send any value $\tilde{\sigma}$. Then, during the computation of β, $\tilde{\mathcal{P}}_2$ transmit the right value σ to $\tilde{\mathcal{S}}$ so that $\mathcal{I}$ is finally π^*-accepted. The hard computational problem involved in the SAV protocol can be easily solved using the private key of the basic protocol. The auxiliary non-repudiation cannot be verified.

4.2 Application to the ZSNS Signature Scheme

In the ZSNS scheme [29], the signer owns public parameters $\big(g,p,e(g, g)\big)$, a public key U and a private key x such that $U = g^x$. To sign a message m, the signer computes the signature $\sigma = \frac{1}{H(m)+x}$, with H a hash function from $\{0, 1\}^*$ into $[\![0, p[\![$. The verifier accepts the signature if $e(\sigma, g^{H(m)}U) = e(g, g)$ (in the generic description, $f(\mathcal{I}, m, r)$ corresponds to $g^{H(m)}U$). This scheme is secure against an adaptive chosen message attack in the random oracle model, under the intractability of the k-CAA problem introduced by Mitsunari et al. [16]:

$$\text{given } g,\ g^x,\ h_0,\ h_1\ \ldots,\ h_k\ (h_i \text{ all different}),\ g^{\frac{1}{h_1+x}}, \ldots,\ g^{\frac{1}{h_k+x}},\ \text{output } g^{\frac{1}{h_0+x}}.$$

We define the *k-Bilinear CAA problem (k-BCAA)* as follows:

$$\text{given } g,\ g^x,\ h_0, \ldots,\ h_k\ (h_i \text{ all different}),\ g^{\frac{1}{h_1+x}}, \ldots,\ g^{\frac{1}{h_k+x}},\ \text{output } e(g, g)^{\frac{1}{h_0+x}}.$$

In appendix B, we prove this problem is harder than the $(k+1)$-Bilinear Diffie Hellman Inversion problem $\big((k + 1)\text{-BDHI}\big)$ introduced by Boneh and Boyen in [2].

Let π denote the ZSNS signature scheme, π^* the generic protocol presented in section 4.1 applied to π, $\mathcal{E}_\pi$ the verification of the equation $e(\sigma, g^{H(m)}U) = e(g, g)$ and $\mathcal{E}_{\pi^*}$ the verification of equation $\beta = e(g, g)^t$.

Theorem 3. *The protocol π^* is a SAV protocol for π, secure against adaptive chosen message attacks in the random oracle model under the intractability of the k-BCAA problem.*

Proof. The auxiliary completeness is already proved in section 4.1.
Auxiliary soundness. Using the model of security of section 2.5, it relies on the following lemma.

Lemma 2. *(With the above notations) Assuming $\mathcal{E}_\pi = 0$, if the server $\tilde{\mathcal{S}}$, communicating with a cheater $\tilde{\mathcal{P}_1}$ which asks q_H queries to the hash oracle and q_S ($< q_H$) queries to the signing oracle, makes $\mathcal{I}$ be π^*-accepted with a probability greater than ε, then the q-BCAA problem, $q \geq q_H + q_S - 1$, can be solved with a probability of success greater than $\frac{(1-\varepsilon')\varepsilon}{q_H}$ and in time $O(\tau)$; with τ denoting the average running time of π^* and ε' the probability of success of a q_S adaptive chosen message attack against the ZSNS signature scheme.*

Let us consider the following random instance of the q-BCAA problem: given g, g^x, h_0, h_1,..., h_q (h_i all different), $g^{\frac{1}{h_1+x}}$,..., $g^{\frac{1}{h_q+x}}$, output $e(g,g)^{\frac{1}{h_0+x}}$.

We now construct an algorithm $\mathcal{A}$ which interacts with $\tilde{\mathcal{P}_1}$ and $\tilde{\mathcal{S}}$, and which solves the above instance of the q-BCAA problem. We assume that $\tilde{\mathcal{P}_1}$ never repeats two same queries to each oracle but can ask a same query to both oracles.

S1 $\mathcal{A}$ prepares a set of answers for the hash oracle: $\mathcal{H} = \{w_0, w_1, \ldots, w_{q_H-1}\} \subset \{h_0, h_1, \ldots h_q\}$ such that $h_0 \in \mathcal{H}$. $\mathcal{A}$ also establishes a list l_H, initially empty, of queries with the corresponding answered hash value.
S2 When $\tilde{\mathcal{P}_1}$ makes a hash query m_i, for $0 \leq i \leq (q_H - 1)$, $\mathcal{A}$ answers w_i and adds the couple (m_i,w_i) in l_H. We denote by $\tilde{m}$ the hash query for which the answer is h_0. If m_i has already been queried to the signing oracle, there exists a couple (m_i,w_i) in l_H and $\mathcal{A}$ answers w_i.
S3 $\mathcal{A}$ also establishes a set S_H, initially empty. When $\tilde{\mathcal{P}_1}$ makes a signing query m_i, two cases are possible: if m_i has already been queried to the hash oracle, there exists a unique couple (m_i,w_i) in l_H; if $m_i = \tilde{m}$, then $\mathcal{A}$ fails, otherwise $\mathcal{A}$ answers $g^{\frac{1}{w_i+x}}$. If m_i has not been queried to the hash oracle, then $\mathcal{A}$ randomly chooses $h_i \in \{h_0, h_1, \ldots, h_q\} \setminus (\mathcal{H} \cup S_H)$, answers $g^{\frac{1}{h_i+x}}$, adds the couple (m_i,h_i) in l_H and adds h_i in S_H.
S4 After making all the queries to the oracles, $\tilde{\mathcal{P}_1}$ outputs a couple (m^*,σ^*). If the message m^* is not equal to $\tilde{m}$ and if (m^*,σ^*) is such that $\mathcal{E}_\pi = 0$, then $\mathcal{A}$ sends to $\tilde{\mathcal{S}}$ the value $\alpha = g^t = g^{(h_0+x)\times\frac{t}{h_0+x}}$ for a random value $t \in [\![0, p[\![$; otherwise, $\mathcal{A}$ fails and then stops.
S5 Finally, $\tilde{\mathcal{S}}$ answers a value β. $\mathcal{A}$ π^*-accepts the couple (m^*,σ^*) if $\beta = e(g,g)^{\frac{t}{h_0+x}}$.

Let n_H denote the number of queries first asked to the random oracle and then asked to the signing oracle. Assuming the hash function behaves like a random oracle, the cheater $\tilde{\mathcal{P}_1}$ cannot distinguish the algorithm $\mathcal{A}$ from a real attack scenario. Moreover the server $\tilde{\mathcal{S}}$ cannot distinguish the value g^t from a value $g^{t'(h_0+x)}$ since t is randomly picked in $\in [\![0, p[\![$. Finally, $\mathcal{A}$ ends if :

1. in step S3, the messages queried to the signing oracle are all different from $\tilde{m}$ which occurs with a probability equal to $\frac{q_H - n_H}{q_H}$,
2. in step S4, the message m^* is equal to $\tilde{m}$ and (m^*,σ^*) is such that $\mathcal{E}_\pi = 0$ which occurs with probability greater than $(1 - \varepsilon')/(q_H - n_H)$,

3. in step S5, $\tilde{\mathcal{S}}$ answers a value β such that $\beta^{t^{-1}} = e(g,g)^{\frac{1}{h_0+x}}$, which happens with a probability greater than ε.

   As a consequence, the probability of success of the algorithm is greater than $\frac{(1-\varepsilon')\varepsilon}{q_H}$.

Computational gain. The hash value $H(m)$ is assumed to be of size $|p|$ so that, using the Remark 2, computing $(Ug^{H(m)})^t$ requires $1.75|p|$ modular multiplications which corresponds to the verifier cost (since $e(g,g)^t$ is precomputed).

Remark 5. It is often complained that a pairing evaluation is too computational expensive to be used in practice. If we consider that such an evaluation is equivalent to the computation of around 4 group exponentiations, then it requires in average $6|p|$ group multiplications.

Using the above remark, and considering that the computation of $Ug^{h(m)}$ requires $1.5|p| + 1$ modular multiplications, we obtain a final cost for the verifier equal to $7.5|p| + 1$ modular multiplications for the basic ZSNS signature scheme. The computation cost in π^* is obviously less than in π such that the computational cost of the verifier is decreased by around 70%. □

5 Conclusion

We have first formalized the concept of a server-aided verification (SAV) protocol, and introduced three properties for such a protocol. Two of them, called auxiliary completeness and auxiliary soundness, are mandatory, while the third one, called auxiliary non-repudiation, must be satisfied only when non-repudiation is required.

In a second time, we have analyzed in this new model two already existing SAV protocols, which both happen to reach the mandatory properties. The first one, proposed by Lim and Lee [14], verifies the auxiliary non-repudiation at the price of requiring the verifier to precompute some values. The second one, initially suggested by Girault and Quisquater [9], is easy to plug into protocols of GPS family, but do not achieve the optional property.

Finally, we have presented a generic SAV protocol for pairing-based schemes, applicable in particular to the Zhang et al. [29] and the Boneh-Boyen [3] signature schemes. Our new method consists in making the server perform the heaviest computation, namely the pairing evaluation, so that the scheme becomes almost as efficient as a ”classical” discrete-logarithm-based signature scheme. But since only the mandatory properties are satisfied, it remains an open problem to find SAV protocols for pairing-based schemes which verify the auxiliary non-repudiation. The Fig. 3 compares the two main characteristics of the different protocols considered in this article.

SAV protocol	Auxiliary non-repudiation	Computational gain
The Lim-Lee method	yes	85%
Server-aided RSA-like GPS scheme	no	95%
Our generic SAV protocol	no	70%

Fig. 3. The different SAV protocols of this article

6 Acknowledgements

The authors wish to thank David Pointcheval and Aline Gouget for valuable and helpful discussions and comments.

References

1. M. Bellare, J. A. Garay, and T. Rabin. Fast Batch Verification for Modular Exponentiation and Digital Signatures. In K. Nyberg, editor, *Advances in Cryptology - Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250. Springer-Verlag, 1998.
2. D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - Eurocrypt '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 382–400. Springer-Verlag, 2004.
3. D. Boneh and X. Boyen. Short Signatures without Random Oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - Eurocrypt '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 382–400. Springer-Verlag, 2004.
4. E. F. Brickell, D. M. Gordon, K. S. McCurley, and D. B. Wilson. Fast Exponentiation with Precomputation (Extended abstract). In R. A. Rueppel, editor, *Advances in Cryptology - Eurocrypt '92*, volume 658 of *Lecture Notes in Computer Science*, pages 200–207. Springer-Verlag, 1993.
5. A. Fiat and A. Shamir. How to Prove Yourself : Practical Solutions to Identification and Signature Problems. In A. M. Odlyzko, editor, *Advances in Cryptology - Crypto '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, 1986.
6. M. Girault. Self-Certified Public Keys. In D. W. Davies, editor, *Advances in Cryptology - Eurocrypt '91*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer-Verlag, 1991.
7. M. Girault and J. C. Paillès. On-line/Off-line RSA-like. In *International Workshop on Coding and Cryptography 2003*, 2003.
8. M. Girault, G. Poupard, and J. Stern. Some Modes of Use of the GPS Identification Scheme. In *3rd Nessie Conference*. Springer-Verlag, 2002.
9. M. Girault and J. J. Quisquater. GQ + GPS = new ideas + new protocols. Eurocrypt '02 - Rump Session, 2002.
10. S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
11. S. Golwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *19^{th} Annual ACM Symposium on the Theory of Computing*, pages 210–217, 1985.
12. L. C. Guillou and J. J. Quisquater. A Practical Zero-knowledge Protocol Fitted to Security Microprocessor Minimizing both Transmission and Memory. In C. G. Günther, editor, *Advances in Cryptology - Eurocrypt '88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer-Verlag, 1988.
13. S. Hohenberger and A. Lysyanskaya. How to Securely Outsource Cryptographic Computations. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 264–282. Springer-Verlag, 2005.
14. C. H. Lim and P. J. Lee. Server (prover/signer)-Aided Verification of Identity Proofs and Signatures. In L. C. Guillou and J. J. Quisquater, editors, *Advances in Cryptology - Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 64–78. Springer-Verlag, 1995.
15. T. Matsumoto, K. Kato, and H. Imai. Speeding up Secret Computations with Insecure Auxiliary Devices. In S. Goldwasser, editor, *Advances in Cryptology - Crypto '88*, volume 403 of *Lecture Notes in Computer Science*, pages 497–506. Springer-Verlag, 1988.
16. S. Mitsunari, R. Sakai, and M. Kasahara. A New Traitor Tracing. *IEICE Trans.*, E85A(2):481–484, 2002.
17. D. M'Raihi and D. Naccache. Couponing Scheme Reduces Computational Power Requirements for DSS Signatures. In *CardTech*, pages 99–104, 1994.
18. P. Q. Nguyen and I. E. Shparlinski. On the Insecurity of a Server-Aided RSA Protocol. In C. Boyd, editor, *Advances in Cryptology - Asiacrypt '01*, volume 2248 of *Lecture Notes in Computer Science*, pages 21–35. Springer-Verlag, 2001.
19. P. Q. Nguyen and J. Stern. The Béguin-Quisquater Server-Aided RSA Protocol from Crypto '95 is not Secure. In K. Ohta and D. Pei, editors, *Advances in Cryptology - Asiacrypt '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 372–379. Springer-Verlag, 1998.

20. B. Pfitzmann and M. Waidner. Attacks on Protocols for Server-Aided RSA Computation. In R. A. Rueppel, editor, *Advances in Cryptology - Eurocrypt '92*, volume 658 of *Lecture Notes in Computer Science*, pages 153–162. Springer-Verlag, 1993.
21. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In U. M. Maurer, editor, *Advances in Cryptology - Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.
22. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In K. Nyberg, editor, *Advances in Cryptology - Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 422–436. Springer-Verlag, 1998.
23. M. O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as Factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology - Laboratory for Computer Science, January 1979.
24. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communication of the ACM*, 21(2):120–126, 1978.
25. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In G. Brassard, editor, *Advances in Cryptology - Crypto '89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer-Verlag, 1990.
26. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
27. J. Shao, R. Lu, and Z. Cao. A New Efficient Identification Scheme Based on the Strong Diffie-Hellman Assumption. In *International Symposium on Future Software Technology*, 2004.
28. M. De Soete and J. J. Quisquater. Speeding Up Smart Card RSA Computations with Insecure Coprocessors. In *Smart Card 2000*, pages 191–198, 1989.
29. F. Zhang, R. Safavi-Naini, and W. Susilo. An Efficient Signature Scheme from Bilinear Pairing and its Applications. In F. Bao, R. H. Deng, and J. Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 277–290. Springer-Verlag, 2004.

A Algorithm from Lemma 1

Let us consider the following algorithm, constructed from [26].

```
$i := 0$
do      $i = i + 1$
        Choose a random tape $\omega_{\tilde{\mathcal{P}}_1}$ and compute $W$
        Send $\tilde{\mathcal{P}}_1$ a random $c_1$ in $[\![0, e[\![$
        Obtain from $\tilde{\mathcal{P}}_1$ the answer $y_1$ and check $\mathcal{E}_\pi$
        Ask the server $\tilde{\mathcal{S}}$ the computation of $Y_1$
until $\big((\mathcal{E}_\pi = 0)$ and $(Y_1^e v^{c_1} = W)\big)$
$j := 0$
do      $j = j + 1$
        Send $\tilde{\mathcal{P}}_1$ a random $c_2 \neq c_1$ in $[\![0, e[\![$
        Obtain from $\tilde{\mathcal{P}}_1$ the answer $y_2$ and check $\mathcal{E}_\pi$
        Ask the server $\tilde{\mathcal{S}}$ the computation of $Y_2$
until $\Big(\big((\mathcal{E}_\pi = 0)$ and $(Y_2^e v^{c_2} = W)\big)$ or $(i == j)\Big)$
if $\big((\mathcal{E}_\pi = 0)$ and $(Y_2^e v^{c_2} = W)\big)$
        then return $(W, c_1, Y_1)$ and $(W, c_2, Y_2)$
        else return FAIL.
```

To analyze this algorithm, we first recall a well-known probabilistic lemma (see [21] for the proof).

Lemma 3. *Let A be an event defined on $X \times Y$ such that $\underset{x,y}{Pr}\big(A(x,y)\big) \geq \varepsilon$ and let $\Omega = \big\{a \in X;$ $\underset{x,y}{Pr}\big(A(x,y)\big) \geq \varepsilon - \alpha\big\}$, then: $\underset{x}{Pr}(x \in \Omega) \geq \alpha$ and $\underset{x}{Pr}\big(x \in \Omega | A(x,y)\big) \geq \alpha/\varepsilon$.*

We assume $\underset{\omega_{\tilde{\mathcal{P}}_1}, \omega_{\mathcal{V}}, \omega_{\tilde{\mathcal{S}}}}{Pr}(\mathcal{V}\ \pi^*\text{-accepts } \mathcal{I} \mid \mathcal{E}_\pi = 0) \geq 1/e + \varepsilon = \varepsilon'$. Let Ω be the set of random tapes $\omega_{\tilde{\mathcal{P}}_1}$ such that $\underset{\omega_{\mathcal{V}}, \omega_{\tilde{\mathcal{S}}}}{Pr}(\mathcal{V}\ \pi^*\text{-accepts } \mathcal{I} \mid \mathcal{E}_\pi = 0) = 1/e + \varepsilon/2$.
Let us now consider the probability of success of this algorithm. Since the GPS RSA-like identification scheme is sound, the probability of success (i.e $\mathcal{E}_\pi = 1$) of a cheater $\tilde{\mathcal{P}}_1$ is less of equal to 1/e so that the probability that ($\mathcal{E}_\pi = 0$) is great or equal than $(1 - 1/e)$.

The probability of success of the first loop is, by definition of $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{P}}_1$, greater than $(1-1/e)\varepsilon'$; and for any integer N, i is equal to N with a probability equal to $\varepsilon'(1-1/e) \times (1-\varepsilon'(1-1/e))^{N-1}$.

The probability that the first loop succeeds with a random tape $\omega_{\tilde{\mathcal{P}}_1}$ lies in Ω is greater than $\varepsilon/(2 \times \varepsilon')$ (using lemma 3).

If $\omega_{\tilde{\mathcal{P}}_1}$ in Ω, the second loop ends with ($\mathcal{E}_\pi = 0$) and ($Y_2^e v^{c_2} = W$) with a probability greater than $\varepsilon/2(1 - 1/e)$; otherwise, $\underset{\omega_{\tilde{\mathcal{P}}_1}, \omega_{\mathcal{V}}, \omega_{\tilde{\mathcal{S}}}}{Pr}(\mathcal{V}\ \pi^*\text{-accepts } \mathcal{I} \mid \mathcal{E}_\pi = 0) \leq \varepsilon'$. Thus, finding this second value c_2 before $j = N$ occurs with a probability greater than $1 - \big(1 - \varepsilon/2(1 - 1/e)\big)^N$. Then, the full probability of success of the algorithm is:

$$\sum_{i=1}^{+\infty} \varepsilon'(1-\frac{1}{e})\Big(1-\varepsilon'(1-\frac{1}{e})\Big)^{i-1} \times \frac{\varepsilon}{2\varepsilon'} \times \left(1-\Big(1-\frac{\varepsilon}{2}(1-\frac{1}{e})\Big)^i\right)$$

which can be rewritten as :

$$\frac{\varepsilon}{2}(1-\frac{1}{e})\left(\sum_{i=0}^{+\infty}\Big(1-\varepsilon'(1-\frac{1}{e})\Big)^i - (1-\frac{\varepsilon}{2}(1-\frac{1}{e}))\sum_{i=0}^{+\infty}\Big((1-\varepsilon'(1-\frac{1}{e}))(1-\frac{\varepsilon}{2}(1-\frac{1}{e}))\Big)^i\right)$$

Using the equality $\sum_{i=0}^{+\infty} x^i = \frac{1}{1-x}$, we obtain

$$\begin{aligned}
&\frac{\varepsilon}{2}(1-\frac{1}{e})\left(\frac{1}{\varepsilon'(1-\frac{1}{e})} - \frac{\Big(1-\frac{\varepsilon}{2}(1-\frac{1}{e})\Big)}{\varepsilon'(1-\frac{1}{e})\Big((1+\frac{\varepsilon}{2\varepsilon'}) - \frac{\varepsilon}{2}(1-\frac{1}{e})\Big)}\right)\\
=&\frac{\varepsilon^2}{4\varepsilon'^2\big((1+\frac{\varepsilon}{2\varepsilon'}) - \frac{\varepsilon}{2}(1-\frac{1}{e})\big)}
\end{aligned}$$

Since $\varepsilon' > \varepsilon$, we obtain $(1 + \frac{\varepsilon}{2\varepsilon'}) - \frac{\varepsilon}{2}(1 - \frac{1}{e}) \leq 1 + \frac{\varepsilon}{2\varepsilon'} \leq (1 + \frac{1}{2}) \leq 3/2$, so that the algorithm succeeds with a probability greater than $\frac{\varepsilon^2}{6\varepsilon'^2}$. The running time of the algorithm is $\frac{2\tau}{(1-1/e)\varepsilon'} \leq \frac{4\tau}{\varepsilon'}$ with τ the average running time of the SAV protocol.

B k-BCAA is a Stronger Problem than $(k+1)$-BDHI

We recall that the k-BCAA problem is defined by:

given g, g^x, h_0, h_1,..., h_k (h_i all different), $g^{\frac{1}{h_1+x}}$,..., $g^{\frac{1}{h_k+x}}$, output $e(g,g)^{\frac{1}{h_0+x}}$.

The $(k+1)$-BDHI problem from [2] is defined by:

$$\text{given } f,\ f^y,\ f^{y^2},\dots,\ f^{y^{k+1}};\ \text{output } e(f,f)^{\frac{1}{y}}.$$

We assume there exists an algorithm which solves the k-BCAA problem and we consider the following given input for the $(k+1)$-BDHI problem: f, f^y, f^{y^2},…, $f^{y^{k+1}}$ for a non zero y.

First, we need to construct a valid input for the k-BCAA problem. Let A_i denote f^{y^i} $\forall i \in [\![0, k+1]\!]$ and let B_i denote $e(f,f)^{y^i}$ $\forall i \in [\![0, 2k-1]\!]$.

Construction of the input of the k-BCAA problem. The method used here is inspired by Mitsunari et al. [16]. Let h_0, h_1,…, h_k be $k+1$ distinct random values in $[\![0, q[\![$ and let $P(Y)$ be the polynomial given by $\prod_{i=1}^{k}(Y - h_0 + h_i)$. This polynomial is of degree k in Y and can be expanded into $P(Y) = \sum_{i=0}^{k} \alpha_i Y^i$.

Let g be the value $f^{P(y)}$ which is obtained by computing $\prod_{i=0}^{k}(A_i)^{\alpha_i}$ and let P_K be the value $g^{-h_0}\prod_{i=0}^{k}(A_{i+1})^{\alpha_i}$ which is also equal to g^{y-h_0}.

Moreover, $\forall j \in [\![1, k]\!]$, let $P_j(Y)$ be the polynomial given by $\prod_{i=1;i\neq j}^{k}(Y - h_0 + h_i)$, so that $(Y - h_0 + h_j)P_j(Y) = P(Y)$; each $P_j(Y)$ can be expanded into $\sum_{i=0}^{k-1}\beta_i^j Y^i$. Then, $\forall j \in [\![1, k]\!]$, we denote by S_j the value $\prod_{i=0}^{k-1}(A_i)^{\beta_i^j}$ which is equal to $f^{P_j(y)} = f^{P(y)/(y-h_0+h_j)} = g^{1/(y-h_0+h_j)}$.

If we finally denote $x = y - h_0$, we then obtain h_0, h_1,…, h_k, g, $P_K = g^x$, $S_1 = g^{1/(x+h_1)}$,…, $S_k = g^{1/(x+h_k)}$ so that we have constructed a valid input for the algorithm solving the k-BCAA problem: we obtain the solution $e(g,g)^{1/(x+h_0)}$ for the current instance of the k-BCAA problem.

Recovering the solution of the $(k+1)$-BDHI. The output value of the k-BCAA problem $e(g,g)^{1/(x+h_0)}$ is equal, by definition, to $e(g,g)^{1/y}$ and more precisely, to $e(f,f)^{P^2(y)/y}$ (we have assumed y is a non zero value).

Let $P'(Y)$ be the rational fraction $P^2(Y)/Y$ which can be written as $\frac{\gamma_{-1}}{Y} + \sum_{i=0}^{2k-1}\gamma_i Y^i$; γ_{-1} is a non zero value equal to $\prod_{i=1}^{k}(h_i - h_0)$. Thus,

$$e(g,g)^{1/(x+h_0)} = e(f,f)^{P^2(y)/y} = e(f,f)^{\frac{\gamma_{-1}}{y} + \sum_{i=0}^{2k-1}\gamma_i y^i} = e(f,f)^{\frac{\gamma_{-1}}{y}} \prod_{i=0}^{2k-1} B_i^{\gamma_i}.$$

As a consequence, we obtain:

$$e(f,f)^{\frac{1}{y}} = \left(e(g,g)^{1/(x+h_0)} \prod_{i=0}^{2k-1} B_i^{-\gamma_i}\right)^{\gamma_{-1}^{-1}}$$