Strengthened Security for Blind Signatures

David Pointcheval*

Abstract. Provable security is a very nice property for cryptographic protocols. Unfortunately, in many cases, this is at the cost of a considerable loss in terms of efficiency. More recently, a new approach to achieve some kind of provable security was explored using the so-called "random oracle model".

Last year, Stern and the author studied the security of blind signatures in this model. We first defined appropriate notions of security for electronic cash purpose, then, we proposed the first efficient and provably secure schemes. Unfortunately, even if our proof prevents a user from spending more coins than he had withdrawn, this is only if the number of withdrawn coins is poly-logarithmically bounded.

In this paper, we propose a generic transformation of those schemes which extends the security even after polynomially many withdrawals. Moreover, this transformation keeps the scheme efficient and so can be used in a secure and efficient anonymous off-line electronic cash system.

Introduction

Since the beginning of modern cryptography with the Diffie-Hellman paper [9], many new schemes have been proposed and many have been broken. Thus, the simple fact that a cryptographic algorithm withstands cryptanalytic attacks for several years is often considered as a kind of validation procedure.

A completely different paradigm is provided by the concept of "provable" security. A significant line of research has tried to provide proofs in the asymptotic framework of complexity theory. Stated in a more accurate way, this approach proposes computational reductions to well established problems, such as factorization, RSA [26], the discrete logarithm problem or any $\mathcal{NP}$-complete problem [12]. Unfortunately, in many cases, provable security is at the cost of a loss in efficiency.

Recently, the scope of these methods has been considerably widened by using a model where concrete cryptographic objects, such as hash functions, are identified with ideal random objects, the so-called "random oracle model" formalized by Bellare and Rogaway [2]. In this model, DES [18] is viewed as a random permutation and SHA [19] as a random function with the appropriate range. This seems to provide a good compromise between security and efficiency.

*GREYC, Département d'Informatique, Université de Caen, Esplanade de la Paix, 14032 Caen Cedex, France and LIENS, École Normale Supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05, France. E-mail: David.Pointcheval@info.unicaen.fr.

Using this model, we follow the study we made [23, 25] with the proposition of several provably secure blind signature schemes. Those protocols are essential ingredients for (revokable) anonymous electronic cash. Our natural notion of security (no "one-more" forgery) was to prevent a user from spending more coins than he had withdrawn. Unfortunately, the proposed schemes [23, 25] provably prevent such a fraud only if the number of withdrawn coins is bounded by a constant, or at least [22] poly-logarithmically bounded.

In this paper, we present a generic transformation one can apply to all our previous schemes and maybe to other ones. The resulting scheme is then provably secure even after polynomially many interactions with the signer.

We now briefly describe the organization of this paper. We first define the so-called "random oracle model" and explain why such a theoretical model can help proving the validity of the design of a cryptographic scheme. We then review the definition of blind signatures and present related results.

In a second part, we propose the idea of the generic transformation with the specific application to the Okamoto-Schnorr blind signature scheme [20, 23]. We then present the proof of the claimed security: if an attacker exists against the resulting scheme, after polynomially many interactions with the signer, then we can construct an attacker against the basic Okamoto-Schnorr blind signature scheme, after logarithmically many interactions with the signer. This latter attack has already been proven impossible unless the discrete logarithm problem is easy to solve.

1 Definitions

1.1 The Random Oracle Model

Many cryptographic schemes use a hash function f. This use of hash functions has originally been motivated by the wish to sign long messages with a single short signature. It was recently realized that hash functions were an essential ingredient for the security of the signature schemes. In order to actually obtain security arguments, while keeping the efficiency of the designs that use hash functions, several authors (e.g. [10], [2, 3] and [24, 23, 25]) have suggested to use the hypothesis that f is actually a random function. We follow this suggestion by using the corresponding model, called the "random oracle model". In this model, the hash function can be seen as an oracle which produces a truly random value for each new query. Of course, if the same query is asked twice, identical answers are obtained. This is precisely the context of relativized complexity theory with "oracles", hence the name. It is argued that proofs in this model ensure security of the overall design of a scheme provided that the hash function has no weakness.

1.2 Blind Signatures

After this presentation of the random oracle model, we review an important cryptographic primitive: blind signatures. We first motivate their use and give some well-known examples. We then recall specific security properties of blind signatures related to the setting of electronic cash and the previous results.

1.2.1 Motivation: Electronic Cash

As early as 1982, David Chaum's [6] pioneering work aimed at creating an electronic version of money. To achieve this goal, he introduced the notions of "coins" and "blind signatures". He claimed that this was the only way to ensure the required *anonymity*: in real life, a coin cannot be easily traced from the bank to the shop, furthermore, two spendings of a same user cannot be linked together. These are two privacy properties of real coins that Chaum wanted to mimic: *untraceability* and *unlinkability.*

He proposed an *electronic coin* to be a number with a certificate (a signature) blindly produced by the Bank; it is withdrawn from the Bank, spent by the user and deposited by the shop. Therefore, all the security of the system relies on the security of the blind signature that we use.

In all the proposed electronic cash schemes, two main blind signature schemes have been used. The first was introduced by Chaum and is a transformation of the RSA signature scheme [26]. And more recently, Brands [4] presented a new scheme derived from the Schnorr signature scheme [27, 28].

The Blind RSA Signature: The generation algorithm produces a large composite number $N = pq$, as well as a pair of related public key e and secret key d. A hash function H is publicly distributed. In order to get the signature of a random number ρ, the user "blinds" it with a random value $r^e \bmod N$, and sends $m = H(\rho)r^e \bmod N$ to the signer. The latter returns a signature σ' of m such that ${\sigma'}^e = m = r^e H(\rho) \bmod N$. A coin is any pair (ρ, σ) which satisfies $\sigma^e = H(\rho) \bmod N$.

The Blind Schnorr Signature: The generation algorithm produces two large prime integers p and q, such that $q \,|\, p-1$, as well as an element g of $\mathbb{Z}_p^\star$ of order q. It also creates a pair of keys, (x, y), where $x \in \mathbb{Z}_q^\star$ is the secret one, and $y = g^{-x} \bmod p$, the public one. The signer publishes y. In order to get the signature of a secret message m, the user asks the signer to initiate a communication. This latter chooses a random $K \in \mathbb{Z}_q^\star$, computes and sends the "commitment" $r = g^K \bmod p$. The user then blinds this value with two random elements $\alpha, \beta \in \mathbb{Z}_q$ into $r' = rg^{-\alpha}y^{-\beta} \bmod p$, computes the value $e' = H(m, r') \bmod q$ and sends the "challenge" $e = e' + \beta \bmod q$ to the signer who returns the value s such that $g^s y^e = r \bmod p$. Finally, the user computes $s' = s - \alpha \bmod q$. This way, the pair (e', s') is a valid Schnorr signature of m since it satisfies the required relation, $e' = H(m, g^{s'} y^{e'} \bmod p)$.

Revokable Anonymity A few years ago [29], an undesirable feature of total anonymity in transactions was considered: perfect crimes (anonymous crimes without leaving any traces and consequently without any risk to be suspected later). Accordingly, a new line of research in electronic cash has investigated "revokable anonymity" [5, 11, 15] which proposes anonymity unless a trusted third party partially revokes it for some established reasons or in view of an obvious fraud (e.g. in case of double-spending). Again, those new schemes rely on the security of blind signature schemes.

1.2.2 Security

A fundamental property for electronic cash systems is the guarantee that a user cannot forge more coins than the Bank gives him. In other words, with ℓ blind signatures of the Bank, the user must not be able to create more than ℓ coins. This form of security was formalized by Stern and the author [23]:

Definition 1 (The $(\ell, \ell+1)$-forgery). For any integer ℓ, an $(\ell, \ell+1)$-forgery comes from an attacker that produces $\ell+1$ signatures after ℓ interactions with the signer Σ.

Definition 2 (The "one-more" forgery). For some integer ℓ, polynomial in the security parameter k, an attacker can obtain $\ell+1$ valid signatures after less than ℓ interactions with the signer. In other words, a *"one-more" forgery* is an $(\ell, \ell+1)$-forgery for some polynomially bounded integer ℓ.

We also envisioned several scenarios depending how the multiple withdrawals are performed: the *sequential attack* and the the *parallel attack*. We now define an intermediate attack:

Definition 3 (The synchronized parallel attack). The attacker can initiate a new interaction whenever he wants, but for each round of the protocol, the interactions indices must be in the same order: the second round of the first interaction must be ended before the second round of the second interaction begins, and so on.

Clearly, this attack is stronger than the sequential attack, but a little less general than the parallel one. Actually, withdrawals can be easily managed this way with a single Bank. In the case of multiple Banks, they have to be synchronized, hence the name.

2 Secure Blind Signature Schemes

We first recall one of our previous schemes with its security results [23]. Then, we present our generic transformation and apply it to this scheme with the proof of security. We only focus on the Okamoto-Schnorr blind signature scheme, but it also works with the Okamoto–Guillou-Quisquater blind signature [20, 23] and our other schemes based on factorization [25].

2.1 The Okamoto-Schnorr Blind Signature Scheme

For a given security parameter k, the initialization algorithm produces two large primes p and q such that $q \mid (p-1)$ and $2^{k-1} < q \leq 2^k$ together with two elements

$g, h \in \mathbb{Z}_p^\star$ of order q. We assume that the hash function f outputs elements in $\mathbb{Z}_q$. The signer Σ chooses a secret key $(r, s) \in (\mathbb{Z}_q^\star)^2$ and publishes the public key, $y = g^{-r}h^{-s} \bmod p$. The protocol by which the user obtains a blind signature of the message m is as follows.

- Σ chooses $(t, u) \in (\mathbb{Z}_t^\star)^2$, computes and sends $a = g^t h^u \bmod p$;
- the user chooses $\beta, \gamma, \delta \in \mathbb{Z}_q$ to blind a into $\alpha = a g^\beta h^\gamma y^\delta \bmod p$. He computes the challenge $\varepsilon = f(m, \alpha)$ and sends $e = \varepsilon - \delta \bmod q$ to Σ;
- the Σ computes $R = t + er \bmod q$ and $S = u + es \bmod q$, and sends a pair (R, S) which satisfies $a = g^R h^S y^e \bmod p$;
- the user computes $\rho = R + \beta \bmod q$ and $\sigma = S + \gamma \bmod q$.

Straightforward computations show that $\alpha = g^\rho h^\sigma y^\varepsilon \bmod p$, with $\varepsilon = f(m, \alpha)$. We provided the following security result [23]:

Lemma 4. *Consider the Okamoto-Schnorr blind signature scheme in the random oracle model. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data. Let us denote by Q and ℓ respectively the number of queries that $\mathcal{A}$ can ask to the random oracle and the number of interaction that $\mathcal{A}$ can ask to the authority. Assume that, within the time bound T, $\mathcal{A}$ produces, with probability $\varepsilon \geq 4Q^{\ell+1}/q$, an $(\ell, \ell+1)$-forgery. Then, the discrete logarithm problem can be solved within time $T' \leq 2T$ and with probability $\varepsilon' \geq 1/4\ell \times (\varepsilon/12\ell Q^{\ell+1})^3$.*

In order to get ε' non-negligible for any polynomial Q, ℓ has to be bounded by a constant. Using the improvement presented in [22] and extending it to an $(\ell, \ell+1)$-forgery after R initiated interactions with the signer (but only ℓ ended ones), we obtain the following theorem:

Theorem 5. *Consider the Okamoto-Schnorr blind signature scheme in the random oracle model. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data. Let us denote by Q, R and ℓ respectively the number of queries that $\mathcal{A}$ can ask to the random oracle, the number of interactions initiated with the authority and the number of interactions completed. Assume that, within the time bound T, $\mathcal{A}$ produces, with probability $\varepsilon \geq 4Q(QR)^\ell/q$, an $(\ell, \ell+1)$-forgery. Then, the discrete logarithm problem can be solved within time $T' \leq 33Q\ell T/\varepsilon$ and with probability $\varepsilon' \geq 1/72\ell^2$.*

Asymptotically, if Q and R can be any polynomials then the number ℓ of interactions has to be poly-logarithmically bounded.

2.2 The Generic Transformation

The idea of the transformation is to impose the user to play honestly, using a kind of “cut-and-choose” technique.

2.2.1 Description

The particular application of the transformation to the Okamoto-Schnorr blind signature scheme is presented on figure 1. But more generally, we introduce a checker $\mathcal{C}$ between the signer Σ and the user $\mathcal{A}$. It will try to detect the cheaters. In fact, on the one hand, we want to verify if the user plays fairly, randomly choosing blinding factors and using them normally: to do so, we ask him to reveal them in order to check the computation of the challenge. But on the other hand, the blinding factors have to remain unknown for the signer, otherwise it is no longer a "blind" signature. Using the notations of the Okamoto-Schnorr scheme, the checker $\mathcal{C}$ works as follows:

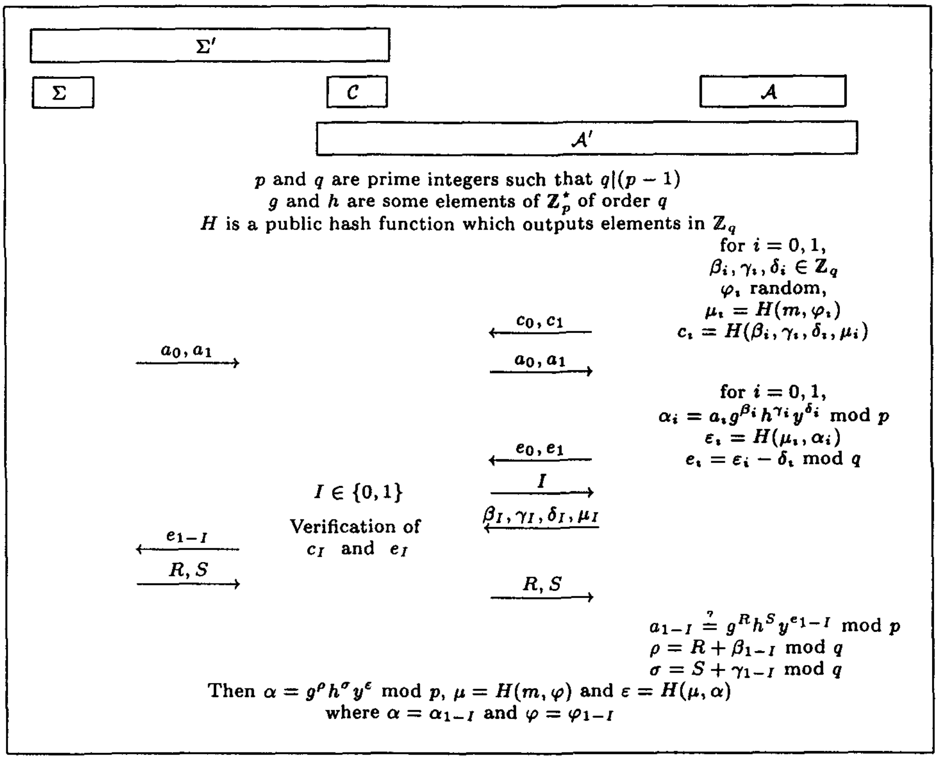

Σ'
Σ
$\mathcal{C}$
$\mathcal{A}$
$\mathcal{A}'$
p and q are prime integers such that $q|(p-1)$
g and h are some elements of $\mathbb{Z}_p^*$ of order q
H is a public hash function which outputs elements in $\mathbb{Z}_q$
for $i = 0, 1$,
$\beta_i, \gamma_i, \delta_i \in \mathbb{Z}_q$
φ_i random,
$\mu_i = H(m, \varphi_i)$
$c_i = H(\beta_i, \gamma_i, \delta_i, \mu_i)$
c_0, c_1
a_0, a_1
a_0, a_1
for $i = 0, 1$,
$\alpha_i = a_i g^{\beta_i} h^{\gamma_i} y^{\delta_i} \bmod p$
$\varepsilon_i = H(\mu_i, \alpha_i)$
$e_i = \varepsilon_i - \delta_i \bmod q$
e_0, e_1
$I \in \{0, 1\}$
I
$\beta_I, \gamma_I, \delta_I, \mu_I$
Verification of c_I and e_I
e_{1-I}
R, S
R, S
$a_{1-I} \overset{?}{=} g^R h^S y^{e_{1-I}} \bmod p$
$\rho = R + \beta_{1-I} \bmod q$
$\sigma = S + \gamma_{1-I} \bmod q$
Then $\alpha = g^\rho h^\sigma y^\varepsilon \bmod p$, $\mu = H(m, \varphi)$ and $\varepsilon = H(\mu, \alpha)$
where $\alpha = \alpha_{1-I}$ and $\varphi = \varphi_{1-I}$

Fig. 1. The Transformed Scheme

- $\mathcal{A}$ commits two tuples of blinding factors $(\beta_i, \gamma_i, \delta_i, \mu_i)$ for $i = 0, 1$ and sends the commitments c_0 and c_1 to $\mathcal{C}$;
- $\mathcal{C}$ initiates two parallel interactions with Σ and receives a_0 and a_1, that it forwards to $\mathcal{A}$;
- $\mathcal{A}$ blinds them respectively in α_0 and α_1, using the blinding factors $(\beta_i, \gamma_i, \delta_i)$ for $i = 0, 1$, computes the challenges ε_0 and ε_1 and the queries e_0 and e_1 that it asks to $\mathcal{C}$;

- At this step, $\mathcal{C}$ checks one of them: it randomly chooses $I \in \{0,1\}$ and asks to verify e_I;
- $\mathcal{A}$ opens c_I, revealing $(\beta_I, \gamma_I, \delta_I, \mu_I)$;
- $\mathcal{C}$ checks the construction of e_I and asks e_{1-I} to Σ which answers (R, S). $\mathcal{C}$ forwards this pair to $\mathcal{A}$;
- $\mathcal{A}$ can then compute ρ and σ such that $\alpha_{1-I} = g^\rho h^\sigma y^{\varepsilon_{1-I}} \bmod p$.

Remark. The random values φ_i are used to perfectly commit the message m into μ_i. The user finally gets a signature of μ_I.

2.2.2 General Technique

From the point of view of $\mathcal{A}$, the collusion of Σ and $\mathcal{C}$ represents the new signer Σ' (see figure 1). Let us assume that $\mathcal{A}$ is an attacker which can perform an $(\ell, \ell+1)$-forgery against Σ', where ℓ is any polynomial, with probability ε. We can remark that during such an attack, $\mathcal{C}$ initiates 2ℓ interactions with Σ but completes only ℓ of them (polynomially many). We will prove that if ε is non-negligible then there exists a machine $\mathcal{S}$ which simulates $\mathcal{C}$, in an indistinguishable way w.r.t. $\mathcal{A}$, with 2ℓ initiated interactions with Σ but only λ completed ones where λ is logarithmically bounded. Furthermore, among the $\ell + 1$ "apparently" valid signatures output by $\mathcal{A}$, there are at least $\lambda + 1$ really valid ones. Then, the collusion of $\mathcal{A}$ and $\mathcal{S}$, denoted by $\mathcal{A}'$, can perform a $(\lambda, \lambda + 1)$-forgery against Σ with λ logarithmically bounded.

2.2.3 Simulator

We now present the simulator $\mathcal{S}$ for a given random oracle f. When it is possible, *i.e.* when the user is honest, the simulator uses a stand-alone simulation, as for honest-verifier zero-knowledge simulations, choosing the expected challenge. But it has also to be able to answer to a cheating user. In this latter case, it needs the help of Σ. We denote by $\mathcal{Q}_i$ and ρ_i respectively the Q queries asked by $\mathcal{A}$ to the hash function H, simulated by $\mathcal{S}$, and the corresponding Q answers. Furthermore, a list Λ_H is constructed and used by $\mathcal{S}$. For any query $\mathcal{Q}$ to H, $\mathcal{S}$ looks if there is a pair $(\mathcal{Q}, \rho) \in \Lambda_H$ for some ρ. If there is such a pair, $\mathcal{S}$ answers ρ. Otherwise, it asks $\rho = f(\mathcal{Q})$, adds the pair $(\mathcal{Q}, \rho)$ to the list Λ_H and answers ρ. This way, $\mathcal{S}$ knows any query that $\mathcal{A}$ asks to H. Moreover, $\mathcal{S}$ has the ability to add some pairs to the list Λ_H, which make H differ a little from f.

1. $\mathcal{S}$ receives the commitments $\{c_i\}_{i=0,1}$ from $\mathcal{A}$. For $i = 0, 1$, $\mathcal{S}$ looks in Λ_H for the query $\mathcal{Q}_{j_i}$ corresponding to c_i. If there is none, we define $j_i = \infty$. But in the positive case, $\mathcal{S}$ learns the blinding factors $\mathcal{Q}_{j_i} = (\beta_i, \gamma_i, \delta_i, \mu_i)$;
2. $\mathcal{S}$ randomly chooses $i \in \{0, 1\}$:
   - **For** $\mathbf{a}_{1-i}$: On the one hand, $\mathcal{S}$ initiates a stand-alone simulation. It randomly chooses u, v and w in $\mathbb{Z}_q$ and defines $a_{1-i} = g^u h^v y^w \bmod p$. We can remark that w is the expected challenge, and to be able to

answer, $\mathcal{S}$ has to make the challenge to be w, by modifying the hash function H. If $j_{1-i} \neq \infty$, with the blinding factors, $\mathcal{S}$ computes the expected query to H, $\alpha_{1-i} = a_{1-i} g^{\beta_{1-i}} h^{\gamma_{1-i}} y^{\delta_{1-i}} \bmod p$, and adds the pair $((\mu_{1-i}, \alpha_{1-i}), w + \delta_{1-i} \bmod q)$ in the list Λ_H. Finally it defines a private variable $E_{1-i} = w$, or $E_{1-i} = \infty$ if $j_{1-i} = \infty$, in order to control the fairness of $\mathcal{A}$;

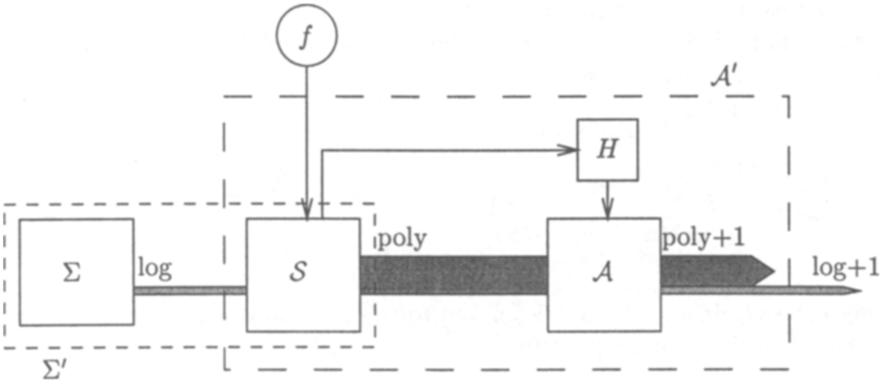
- **For** $\mathbf{a_i}$: On the other hand, $\mathcal{S}$ is a mediator between Σ and $\mathcal{A}$. This will help it to be able to answer to any query, using Σ, from a possibly cheater $\mathcal{A}$. Then it asks for a_i to the signer Σ. To make both parts symmetric, we have to know if $\mathcal{A}$ is also trying to cheat. But here, we need not modify H since $\mathcal{S}$ will be able to answer any query using Σ. If $j_i \neq \infty$, with the blinding factors, $\mathcal{S}$ computes the expected query to the random oracle, $\alpha_i = a_i g^{\beta_i} h^{\gamma_i} y^{\delta_i} \bmod p$, and then computes the expected challenge $e_i = H(\mu_i, \alpha_i) - \delta_i = f(\mu_i, \alpha_i) - \delta_i \bmod q$. As above, it defines a private variable $E_i = e_i$, or $E_i = \infty$ if $j_i = \infty$, in order to control the fairness of $\mathcal{A}$;

Finally, $\mathcal{S}$ sends a_0 and a_1 to $\mathcal{A}$;

3. $\mathcal{A}$ sends the challenges e_0 and e_1;
4. If $(e_0, e_1) = (E_0, E_1)$ then $\mathcal{A}$ asks the expected challenges and so had played honestly in both parts: $\mathcal{S}$ can end the stand-alone simulation and so will ask to verify c_i, then it defines $I = i$. Otherwise, $\mathcal{A}$ had tried to cheat, and so $\mathcal{S}$ may not be able to end the stand-alone simulation and it will need the help of Σ. It then lets $I = 1 - i$. $\mathcal{S}$ sends I to $\mathcal{A}$;
5. $\mathcal{A}$ answers $\beta', \gamma', \delta', \mu'$;
6. $\mathcal{S}$ checks whether $c_I = f(\beta', \gamma', \delta', \mu')$, or simply $Q_{j_I} = (\beta', \gamma', \delta', \mu')$. If not, it stops the game. Else
   - if $I = i$ then $\mathcal{S}$ replies u, v;
   - otherwise, it asks e_{1-I} to Σ which answers R, S. $\mathcal{S}$ forwards R, S.

After ℓ such interactions, $\mathcal{A}$ outputs $\ell + 1$ valid signatures (w.r.t. H), m_i, α_i, ρ_i, σ_i, φ_i such that $\alpha_i = g^{\rho_i} h^{\sigma_i} y^{\varepsilon_i} \bmod p$, $\mu_i = H(m_i, \varphi_i)$ and $\varepsilon_i = H(\mu_i, \alpha_i)$, for $i = 1, \ldots, \ell + 1$. But only few signatures are really valid (w.r.t. f) and furthermore satisfy $\varepsilon_i = f(\mu_i, \alpha_i)$. We will denote by λ the number of completed interactions with Σ. We can remark that the number of initiated interactions with Σ is equal to ℓ. Now, we want to prove that the collusion of $\mathcal{A}$ and $\mathcal{S}$ performs a $(\lambda, \lambda{+}1)$-forgery against Σ with λ logarithmically bounded (see figure 2). We require the three following properties:

- any $\mathcal{A}$ cannot distinguish $\mathcal{S}$ from $\mathcal{C}$;
- if $\mathcal{A}$ outputs $\ell + 1$ valid signatures (w.r.t. H), there are at least $\lambda + 1$ really valid signatures (w.r.t f).
- if ε is non-negligible, $\mathcal{S}$ completes only logarithmically many interactions with Σ;



•	Resulting scheme	Σ': signer	$\mathcal{A}$: attacker
•	Basic scheme	Σ: signer	$\mathcal{A}'$: attacker
•	$\mathcal{S}$ Simulator		
•	f random oracle		
•	H $\mathcal{S}$-controlled hash function		

Fig. 2. Reduction

Indistinguishability. The first thing we have to verify is that the attacker cannot remark that we have replaced $\mathcal{C}$ and f by $\mathcal{S}$ and H:

- first, it is clear that a_0 and a_1 sent by $\mathcal{S}$ have exactly the same distribution as those sent by Σ and so by $\mathcal{C}$;
- secondly, with very high probability, H seems to be a random oracle. In fact, since its answers are either the answer of f, the really random oracle or $w + \delta \bmod q$, where w is randomly chosen, the outputs are random values in $\mathbb{Z}_q$. But to be like a random oracle, H must answer the same value for identical queries. A collision of queries may occur when $\mathcal{S}$ adds a pair $((\mu, \alpha), w + \delta)$ in Λ_H. This query (μ, α) may have already been asked to H by the attacker $\mathcal{A}$. But the probability of such an event is less than Q/q, for any pair. So that, the probability of collision of queries is less than $Q\ell/q$;
- finally, the challenge I asked by $\mathcal{S}$ is equal to $i \oplus v$, where i is a really random bit and v is equal to 1 if and only if $(e_0, e_1) = (E_0, E_1)$, which is totally independent of i. Then I really looks like flipping a coin.

We have assumed that $\mathcal{A}$ is an attacker against Σ' and performs an $(\ell, \ell+1)$-forgery with probability ε: $\Pr_{\omega, \omega_\Sigma, \omega_\mathcal{C}, f, \mathcal{I}}[\mathcal{A}^{\Sigma, \mathcal{C}, f} \text{ succeeds}] = \varepsilon$, where we denote by ω, ω_Σ and $\omega_\mathcal{C}$ respectively the random tapes of $\mathcal{A}$, Σ and $\mathcal{C}$, by f the random oracle and by $\mathcal{I}$ the list of the challenges I. As said in [23], we can assume that, for any valid output signature $(m_i, \alpha_i, \rho_i, \sigma_i, \varphi_i)$ which satisfies $\alpha_i = g^{\rho_i} h^{\sigma_i} y^{\varepsilon_i} \bmod p$, $\mu_i = H(m_i, \varphi_i)$ and $\varepsilon_i = H(\mu_i, \alpha_i)$, the attacker has asked the queries (m_i, φ_i) and (μ_i, α_i) to the random oracle during the attack. Otherwise the probability of success would be less than $2(\ell+1)/q$. Furthermore, from the indistinguishability result,

$$\Pr_{\omega,\omega_\Sigma,\omega_S,f,\mathcal{I}}\left[\begin{array}{l}\mathcal{A}^{\Sigma,\mathcal{S},f} \text{ succeeds} \\ (\forall i)(\exists j_1, j_2) \\ \left(\begin{array}{l} \mathcal{Q}_{j_1} = (m_i, \varphi_i) \\ \mathcal{Q}_{j_2} = (\mu_i, \alpha_i) \end{array}\right) \\ \text{and no collisions of queries}\end{array}\right] \geq \varepsilon - \frac{2(\ell+1)}{q} - \frac{Q\ell}{q} \geq \varepsilon - 2\frac{Q\ell}{q},$$

where the $\mathcal{Q}_i$ denote the queries asked to the random oracle. Let us denote by $\mathcal{G}$ the set of those successful tuples $(\omega, \omega_\Sigma, \omega_S, f, \mathcal{I})$.

The number of valid signatures is greater than $\lambda + 1$. To prove this statement, we can upper bound the number of invalid signatures. Assume that for some i, $\varepsilon_i = H(\mu_i, \alpha_i) \neq f(\mu_i, \alpha_i)$. Then $((\mu_i, \alpha_i), \varepsilon_i)$ is a pair added by $\mathcal{S}$ to Λ_H: $\varepsilon_i = w + \delta$ and $\alpha_i = a g^\beta h^\gamma y^\delta \bmod p$, where $a = g^u h^v y^w \bmod p$. Then

$$g^{\rho_i - \beta} h^{\sigma_i - \gamma} = a y^{-w} = g^u h^v \bmod p.$$

This means that

- either $\mathcal{A}$ received u and v from $\mathcal{S}$;
- or $\mathcal{A}$ had computed ρ_i and σ_i from ay^{-w}. In this case, with probability greater than $1 - 1/q$, $\rho_i \neq u + \beta$, then one has $g^u h^v = g^{\rho_i - \beta} h^{\sigma_i - \gamma} \bmod p$, which provides the discrete logarithm of h relatively to g.

With overwhelming probability, greater than $1 - \ell/q$, in case of forgery, for any i, the inequality $\varepsilon_i = H(\mu_i, \alpha_i) \neq f(\mu_i, \alpha_i)$ implies that $\mathcal{S}$ has completely simulated the signer. Then,

$$\#\{i \,|\, \varepsilon_i = f(\mu_i, \alpha_i)\} = \ell + 1 - \#\{i \,|\, \varepsilon_i \neq f(\mu_i, \alpha_i)\} \geq \ell + 1 - (\ell - \lambda) \geq \lambda + 1.$$

We will denote by $\mathcal{G}'$ the subset of the tuples $(\omega, \omega_\Sigma, \omega_S, f, \mathcal{I}) \in \mathcal{G}$ which provide an $(\lambda, \lambda+1)$-forgery against Σ, and without having found any collision for f, in the classical sense. This latter requirement will be useful in the following. Let us recall that the probability to find a collision for a random function f is less than $Q^2/2q$ after Q queries. Then

$$\Pr[\mathcal{G}'] = \nu \geq \left(\varepsilon - 2\frac{\ell Q}{q}\right)\left(1 - \frac{\ell}{Q}\right) - Q^2/2q \geq \varepsilon - \frac{(Q+1)(2\ell + Q/2)}{q}.$$

This is greater than $\varepsilon/2$ if $\varepsilon \geq 6Q^2/q$, since $\ell < Q$.

The number of interactions with Σ is logarithmically bounded. We have shown that the collusion of $\mathcal{A}$ and $\mathcal{S}$ provides a "one-more" attacker $\mathcal{A}'$ after λ interactions with Σ. We would like λ to be logarithmically bounded to derive a contradiction. This is for this part of the proof that we need the attacker to perform a synchronized parallel attack, in order to have an execution binary tree: we cannot let a cheater postpone an interaction for which he cannot correctly open the commitment c_I. We first enounce a lemma about execution binary trees, whose proof can be found in the appendix A.

Definition 6. An *execution binary tree* T of depth d is a binary tree whose leaves are all at depth d.
A *single node* in an execution binary tree is an internal node (not a leaf) with exactly one son.

Lemma 7. *For any execution binary tree T of depth d, if we denote by $P(T,i)$ the number of paths (from the root to a leaf) containing exactly i single nodes, then*

$$\sum_i 2^i \cdot P(T,i) = 2^d.$$

By definition, $\Pr[\mathcal{G}'] \geq \nu$. Let us define the set

$$\Omega = \left\{ (\omega, \omega_\Sigma, \omega_S, f, \mathcal{I}) \mid \Pr_{\mathcal{I}'}[(\omega, \omega_\Sigma, \omega_S, f, \mathcal{I}') \in \mathcal{G}'] \geq \frac{\nu}{2} \right\},$$

then $\Pr[\Omega] \geq \nu/2$ and $\Pr[\Omega \mid \mathcal{G}'] \geq 1/2$, as proven in appendix B.

Let $(\omega, \omega_\Sigma, \omega_S, f, \mathcal{I}) \in \Omega$ and $T = T(\omega, \omega_\Sigma, \omega_S, f)$ be the execution tree, for $(\omega, \omega_\Sigma, \omega_S, f)$ fixed, for all the possible sequences of queries $\mathcal{I}$. Then T is a binary execution tree of depth ℓ, with a number of leaves, denoted by N, greater then $2^\ell \nu/2$. Furthermore, since we have restricted the attacker to synchronized parallel attacks, for any path $\mathcal{I}$ in T, the node at depth $d-1$ corresponds to the query I of the d^{th} signature. Furthermore, a double node in this tree means that $\mathcal{A}$ can answer both queries, $I = 0$ and $I = 1$ at the step 5. This is possible if either $(e_0, e_1) = (E_0, E_1)$ or $\mathcal{A}$ has found a collision for f, which has been excluded. Therefore, when $\mathcal{S}$ asks help of Σ, there is a single node in the tree: λ is less than the number of single nodes in the path.

For any s, the number $N(s)$ of paths with at least $s+1$ single nodes can be upper bounded:

$$2^\ell = \sum_{i=0}^{i=\ell} P(T,i) \cdot 2^i \geq \sum_{i=s+1}^{i=\ell} P(T,i) \cdot 2^i \geq 2^{s+1} \sum_{i=s+1}^{i=\ell} P(T,i) = 2^{s+1} N(s).$$

Then, we can upper bound the probability for λ to be greater than s:

$$\Pr_{\mathcal{I}}[\lambda > s \mid \mathcal{G}'] \leq N(s)/N \leq 2^{-s}/\nu.$$

Let us define $s = \lceil \log(4/\varepsilon) \rceil$. We then can evaluate the probability to obtain a forgery with less than s calls to Σ:

$$\begin{aligned} \Pr[\mathcal{G}' \;\&\; \lambda \leq s] &\geq \Pr[\mathcal{G}' \;\&\; \Omega \;\&\; \lambda \leq s] \\ &\geq \Pr[\lambda \leq s \mid \mathcal{G}' \cap \Omega] \times \Pr[\Omega \mid \mathcal{G}'] \times \Pr[\mathcal{G}'] \\ &\geq (1 - 2^{-s}/\nu) \times \nu/2 \geq 1/2 \times \varepsilon/4 \geq \varepsilon/8. \end{aligned}$$

As a consequence, in case of forgery, with probability greater than one over 8, the number of calls to Σ is less than $\log(8/\varepsilon)$.

Characteristics of the new attacker $\mathcal{A}'$. We have then constructed an attacker $\mathcal{A}'$ which performs a $(\lambda, \lambda+1)$-forgery with $\lambda \leq \log(8/\varepsilon)$ with probability ε' greater than $\varepsilon/8$, within time bound T and after less than ℓ initiated interactions with Σ.

If $\varepsilon^{2\log Q+1} \geq 32Q^7/q$, then $\varepsilon' \geq 4Q(Q\ell)^\lambda/q$ and moreover $\varepsilon \geq 6Q^2/q$. Consequently, using theorem 5, the discrete logarithm problem can be solved within a time bound $T'' \leq 33\log(8/\varepsilon)QT/\varepsilon$ and with probability $\varepsilon'' \geq (8.5\log(8/\varepsilon))^{-2}$, if $\lambda \leq \log(8/\varepsilon)$. This latter condition occurs with probability at least 1/8 when we are in $\mathcal{G}'$. The global probability of success is therefore greater than $(25\log(8/\varepsilon))^{-2}$, hence the theorem.

Theorem 8. *Consider the transformed scheme in the random oracle model under a synchronized parallel attack. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data. Let us denote respectively by Q and ℓ the number of queries that $\mathcal{A}$ can ask to the random oracle and the number of signatures that $\mathcal{A}$ can get from the signer. Assume that, within the time bound T, $\mathcal{A}$ produces, with probability $\varepsilon \geq (32Q^7/q)^{1/(2\log Q+1)}$, an $(\ell, \ell+1)$-forgery. Then, the discrete logarithm problem can be solved within time $T' \leq 33\log(8/\varepsilon)QT/\varepsilon$ and with probability $\varepsilon' \geq 1/(25\log(8/\varepsilon))^2$.*

Conclusion

We have presented a generic slight transformation which can be applied to many witness-indistinguishable-based blind signature schemes [20, 23, 25]. Furthermore, this transformation is proven to turn the exponential contribution of the number ℓ of interactions with the signer into a polynomial one in the constraints for the reduction. Consequently, with this transformation, the schemes provably prevent "one-more" forgeries even after polynomially many interactions with the signer.

We give a partial but practical answer to our open problem [23] about the possibility to provide a blind signature scheme secure after polynomially many interactions with the signer. Damgård [8], Pfitzmann and Waidner [21], and more recently Juels, Luby and Ostrovsky [16], had answered this question positively in a theoretical but inefficient way. Indeed, they all use provably secure signature schemes [1, 14, 17] together with secure two-party computation protocols [13, 7] in the standard security model. The originality of this paper is to present efficient schemes which prevent any "one-more" forgery against synchronized parallel attacks in the random oracle model.

References

[1] M. Bellare and S. Micali. How To Sign Given Any Trapdoor Function. In *Crypto '88*, LNCS 403, pages 200–215. Springer-Verlag, 1989.

[2] M. Bellare and P. Rogaway. Random Oracles are Practical: a paradigm for designing efficient protocols. In *Proc. of the 1st CCCS*, pages 62–73. ACM press, 1993.

[3] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416. Springer-Verlag, 1996.

[4] S. A. Brands. Untraceable Off-line Cash in Wallets with Observers. In *Crypto '93*, LNCS 773, pages 302–318. Springer-Verlag, 1994.

[5] J. Camenisch, U. Maurer, and M. Stadler. Digital Payment Systems with Passive Anonymity-Revoking Trustees. In *ESORICS '96*, LNCS 1146. Springer-Verlag, 1996.

[6] D. Chaum. Blind Signatures for Untraceable Payments. In *Crypto '82*, pages 199–203. Plenum, NY, 1983.

[7] D. Chaum, I. B. Damgård, and J. van de Graaf. Multiparty Computations ensuring Privacy of each Party's Input and Correctness of the Result. In *Crypto '87*, LNCS 293. Springer-Verlag, 1988.

[8] I. B. Damgård. Payment Sytems and Credential Mechanisms with Provable Security against Abuse by Individuals. In *Crypto '88*, LNCS 403, pages 328–335. Springer-Verlag, 1989.

[9] W. Diffie and M. E. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, volume IT–22, no. 6, pages 644–654, November 1976.

[10] A. Fiat and A. Shamir. How to Prove Yourself: practical solutions of identification and signature problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, 1987.

[11] Y. Frankel, Y. Tsiounis, and M. Yung. "Indirect Disclosure Proof": Achieving Efficient Fair Off-Line E-Cash. In *Asiacrypt '96*, LNCS 1163, pages 286–300. Springer-Verlag, 1996.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of* NP*-Completeness.* Freeman, New-York, 1979.

[13] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game. In *Proc. of the 19th STOC*, pages 218–229. ACM Press, 1987.

[14] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM journal of computing*, 17(2):281–308, April 1988.

[15] M. Jakobsson and M. Yung. Revokable and Versatile Electronic Money. In *Proc. of the 3rd CCCS*, pages 76–87. ACM press, 1996.

[16] A. Juels, M. Luby, and R. Ostrovsky. Security of Blind Digital Signatures. In *Crypto '97*, LNCS 1294, pages 150–164. Springer-Verlag, 1997.

[17] M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In *Proc. of the 21st STOC*, pages 33–43. ACM Press, 1989.

[18] National Bureau of Standard U.S. *Data Encryption Standard*, 1977.

[19] NIST. Secure Hash Standard (SHS). Federal Information Processing Standards PUBlication 180–1, April 1995.

[20] T. Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In *Crypto '92*, LNCS 740, pages 31–53. Springer-Verlag, 1992.

[21] B. Pfitzmann and M. Waidner. How to Break and Repair a "Provably Secure" Untraceable Payment System. In *Crypto '91*, LNCS 576, pages 338–350. Springer-Verlag, 1992.

[22] D. Pointcheval. *Les Preuves de Connaissance et leurs Preuves de Sécurité.* PhD thesis, Université de Caen, December 1996.

[23] D. Pointcheval and J. Stern. Provably Secure Blind Signature Schemes. In *Asiacrypt '96*, LNCS 1163, pages 252–265. Springer-Verlag, 1996.

[24] D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398. Springer-Verlag, 1996.

[25] D. Pointcheval and J. Stern. New Blind Signatures Equivalent to Factorization. In *Proc. of the 4th CCCS*, pages 92–99. ACM press, 1997.

[26] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[27] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, 1990.

[28] C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[29] S. von Solms and D. Naccache. On Blind Signatures and Perfect Crimes. *Computers & Security*, 11:581–583, 1992.

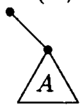
A Proof of lemma 7

We can easily prove this lemma by induction over the depth d:

- $d = 0$: T is the tree with only one leaf, the root. Then, $P(T,0) = 1$ and for any $i \geq 1$, $P(T,i) = 0$. So, $\sum_i 2^i \cdot P(T,i) = 1 = 2^0$.

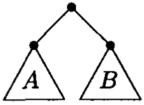
  Remark. We can easily prove this lemma for any full binary tree of depth d: for such a full tree, $P(T,0) = 2^d$ and for any $i \geq 1$, $P(T,i) = 0$, so, $\sum_i 2^i \cdot P(T,i) = 1 \cdot P(T,0) = 2^d$.

- T is a tree of depth d. Two cases can appear:
  - The root has only one son, the subtree A of depth $d-1$. Then, for any i, $P(T,i) = P(A,i-1)$. Therefore,



$$\begin{aligned}\textstyle\sum_i 2^i \cdot P(T,i) &= \textstyle\sum_i 2^i \cdot P(A,i-1)\\ &= 2 \times \textstyle\sum_i 2^i \cdot P(A,i)\\ &= 2 \times 2^{d-1} = 2^d\end{aligned}$$

  - The root has two sons, the subtree A and B of depth $d-1$. Then, for any i, $P(T,i) = P(A,i) + P(B,i)$. Therefore,



$$\begin{aligned}\textstyle\sum_i 2^i \cdot P(T,i) &= \textstyle\sum_i 2^i \cdot (P(A,i) + P(B,i))\\ &= \textstyle\sum_i 2^i \cdot P(A,i) + \sum_i 2^i \cdot P(B,i)\\ &= 2 \times 2^{d-1} = 2^d\end{aligned}$$

B Size of Ω

Let us define the set

$$\Omega = \left\{(\omega, \omega_\Sigma, \omega_S, f, \mathcal{I}) \,\middle|\, \Pr_{\mathcal{I}'}[(\omega, \omega_\Sigma, \omega_S, f, \mathcal{I}') \in \mathcal{G}'] \geq \frac{\nu}{2}\right\}.$$

Let us assume that $\Pr[\Omega] < \nu/2$, then

$$\nu \leq \Pr[\mathcal{G}'] = \Pr[\mathcal{G}' \mid \Omega] \cdot \Pr[\Omega] + \Pr[\mathcal{G}' \mid \bar{\Omega}] \cdot \Pr[\bar{\Omega}] < \nu/2 + \nu/2 = \nu,$$

which implies a contradiction. Furthermore, using Bayes' law

$$\Pr[\Omega \mid \mathcal{G}'] = 1 - \Pr[\bar{\Omega} \mid \mathcal{G}'] = 1 - \Pr[\mathcal{G}' \mid \bar{\Omega}] \cdot \Pr[\bar{\Omega}] / \Pr[\mathcal{G}'] \geq 1 - \nu/2 \times 1/\nu = 1/2.$$