

# Cryptanalysis of a Partially Blind Signature Scheme

## or *How to make \$100 bills with \$1 and \$2 ones*

Gwenaëlle Martinet<sup>1</sup>, Guillaume Poupard<sup>1</sup>, and Philippe Sola<sup>2</sup>

<sup>1</sup> DCSSI Crypto Lab, 51 boulevard de La Tour-Maubourg  
F-75700 Paris 07 SP, France

{Gwenaëlle.Martinet,Guillaume.Poupard}@sgdn.pm.gouv.fr

<sup>2</sup> psola78@hotmail.com

**Abstract.** Partially blind signature scheme is a cryptographic primitive mainly used to design efficient and anonymous electronic cash systems. Due to this attractive application, some researchers have focused their interest on it. Cao, Lin and Xue recently proposed such a protocol based on RSA. In this paper we first show that this protocol does not meet the anonymous property since the bank is able to link a signature with a user. We then present a cryptanalysis of this scheme. In practical applications, a consequence would be the possibility for an attacker to forge, for example, valid \$100 bills after the withdrawal of only two bank notes of \$1 and \$2.

**Keywords:** Cryptanalysis, partially blind signature, electronic cash.

## 1 Introduction

Blind signatures are variants of digital signature schemes for which the signer does not learn the message he actually signs. At first sight, such a primitive is surprising but it enables the design of electronic cash or voting system which protect the anonymity of users. This idea was initially introduced by Chaum [4] but many work has been done on this topic since then [5, 6].

In a very simple approach, an e-cash system can be described in the following way: first, a user withdraws a bill from the bank. This means that the bank digitally signs a message and decreases the balance of the user's account. Then, the user gives the electronic bill to the merchant when he wants to pay. Finally, the merchant gives the bill back to the bank to be refunded. Since the bill is electronically signed by the bank, the merchant and the bank itself can check its validity. However the double-spending of a bill can be avoided only in an online setting. Otherwise, the only solution is to trace dishonest users.

In order to make the money anonymous, a nice solution is to use blind signatures during withdrawal; in such a way, the bank is not able to link a withdrawn bill and an electronic banknote given by a merchant for refunding. We refer to [8] for an exhaustive bibliography on this topic. An elegant solution to cope with

the necessity of checking the correctness of messages during blind signature is to use so-called *partially* blind signatures. They have been introduced by Abe and Fujisaki [1] and further formalized by Abe and Okamoto [2]. In the e-cash scenario, the bank signs messages made of two parts; some information such as the value of the bill and the expiration date are visible by the signer but other data such as serial numbers are still invisible and blindly signed.

Recently, Cao, Lin and Xue [3] proposed such a partially blind signature protocol. We describe this scheme in section 2. Then, we show that the scheme does not fulfill the requirement on the anonymity of the user. We also show that this scheme is not secure since an attacker can generate valid signatures under the realistic assumption that concurrent signatures are allowed. In the e-cash scenario, the consequences could be dramatic; for example, it is possible to withdraw two small bills of, let's say, \$1 and \$2 and then to generate, without any communication with the bank and consequently without any modification of the account balance, a \$100 bank note which has apparently been correctly signed by the bank. Furthermore, this cheating cannot be detected by the bank if the small bills are never spent.

## 2 The Cao-Lin-Xue partially blind signature scheme

Let us now remind the partially blind signature scheme proposed by Cao, Lin and Xue [3]. We consider two parties, a user and a signer; at the end of the protocol, the user obtains a valid signature issued by the signer for a message  $m$  of its choice and for a string  $a$  agreed by both the user and the signer.

The protocol uses a one-way hash function  $H$  that generates  $k$ -bit hash values. We also need a variant  $\tau$  of  $H$  defined by  $\tau(a) = 2^k + H(a)$ .

The signer has a standard RSA signature key pair with large public exponent. More precisely, let  $p$  and  $q$  be two primes such that the factorization of  $n = p \times q$  is intractable. The public exponent  $e$  is an integer larger than  $2^{k+1}$  and relatively prime with  $(p-1) \times (q-1)$ . The private exponent  $d$  is the inverse of  $e$  modulo  $(p-1) \times (q-1)$ . The signer's public key is  $(e, n)$  and the private key is  $(d, n)$ .

Let  $a$  be a string agreed by both the user and the signer. For example it may encode the value of a bill and an expiration date. This information is known by the signer and should not help to further reveal the anonymity of the user. We also consider a message  $m$  chosen by the user and not revealed to the signer; this message is blindly signed by the signer who must not learn information about  $m$ . The partially blind signature of the pair  $(a, m)$  is performed as described in figure 1. The following notations are used:

- $x \in_U X$  means that  $x$  is randomly chosen in the set  $X$  using a uniform distribution,
- $\mathbb{Z}_n$  denotes the set of integers modulo  $n$ ,
- $\mathbb{Z}_n^*$  denotes the multiplicative group of invertible elements of  $\mathbb{Z}_n$ ,
- all the computations are performed modulo the RSA modulus  $n$ ,
- $H(x||y)$  means that the hash value of the concatenation of the binary strings representing data  $x$  and  $y$  is computed using the hash function  $H$ .

User		Signer
Private input : a message $m$		
Choose a string $a$	$\xrightarrow{a}$	check the validity of $a$ $x \in_U \mathbb{Z}_n^*$
	$\xleftarrow{y}$	$y = x^e \bmod n$
$r \in_U \mathbb{Z}_n^*, u \in_U \mathbb{Z}_n^*$		
$\alpha = r^e u H(m    u^e y \bmod n) \bmod n$	$\xrightarrow{\alpha}$	
	$\xleftarrow{t, x}$	$t = (\alpha x)^{-d\tau(a)} \bmod n$
$c = ux \bmod n$		
$s = r^{\tau(a)} t \bmod n$		
$(s, c, a)$ is a signature of the message $m$		

**Fig. 1.** The Cao, Lin and Xue protocol from [3]

A signature  $(s, c, a)$  for a message  $m$  is valid if

$$s^e (H(m || c^e) c)^{\tau(a)} = 1 \bmod n \quad (1)$$

The correctness of the protocol can be easily checked since, if both the user and the signer are honest and follow the protocol:

$$\begin{aligned}
s^e (H(m || c^e) c)^{\tau(a)} &= \left( r^{e\tau(a)} t^e \right) H(m || u^e x^e)^{\tau(a)} u^{\tau(a)} x^{\tau(a)} \\
&= r^{e\tau(a)} \left( \alpha^{-ed\tau(a)} x^{-ed\tau(a)} \right) H(m || u^e x^e)^{\tau(a)} u^{\tau(a)} x^{\tau(a)} \\
&= r^{e\tau(a)} r^{-ed\tau(a)} u^{-\tau(a)} H(m || u^e y)^{-\tau(a)} H(m || u^e x^e)^{\tau(a)} u^{\tau(a)} \\
&= 1 \bmod n
\end{aligned}$$

### 3 Anonymity in the Cao-Lin-Xue scheme

The anonymity property ensures that a user and a valid signature cannot be linked, even by the signer. This property, also known as blindness property, has been formalized in [5]. Informally, it guarantees that an attacker cannot deduce from a target signature the transcript from which it is issued. Even the signer should not be able to trace a signature, *i.e.* the knowledge of the private signing key should not help to break this property.

Let us now consider the Cao-Lin-Xue scheme. In [3] the blindness property is considered. However the proof given is clearly wrong since the signer is able to link a signature with one of his transcript.

Using the notations of figure 1, let  $\{(a, y_i, \alpha_i, t_i, x_i)\}$  be the set of transcripts between a signer and all the users. Let  $(s, c, a)$  a target signature for a message  $m$ , computed during the  $k$ -th transcript,  $(a, y_k, \alpha_k, t_k, x_k)$ . The signer's goal is to link this signature with one of the users. We suppose the value  $a$  is the same for all of the users, otherwise the signer trivially associates the signature with the

corresponding transcript. For all the values  $i$ , since the signer knows the private signature key, he can compute the following values:

$$u = \frac{c}{x_i} \pmod n \quad \text{and} \quad r = \left( \frac{\alpha_i}{u \times H(m||u^e y_i)} \right)^d \pmod n$$

If  $i = k$ , then the values  $u$  and  $r$  computed are those used during the signature generation, *i.e.*  $u = u_k$  and  $r = r_k$ . Otherwise, these values are random ones, linked neither with  $u_i$  and  $r_i$  nor with  $u_k$  and  $r_k$ . The signer then computes the value  $s = r^{\tau(a)} t_i$  and checks if  $s = s_i$ . If the equality holds, then  $k = i$  and the signer can link the target signature with a user. Otherwise, the transcript is not the one used to generate the target signature and the signer tries another one.

## 4 Cryptanalysis of Cao-Lin-Xue scheme

### 4.1 Some basic ideas

A strong goal for an attacker may be to forge signatures, *i.e.* to generate valid signatures that has not been actually produced by the signer. Let us assume that we know two signatures  $(s_1, c_1, a_1)$  and  $(s_2, c_2, a_2)$  of the same message  $m$  for two different strings  $a_1$  and  $a_2$ . In order to give the intuition of the attack, we analyze the consequences of the equality of  $c_1$  and  $c_2$ . Let  $c = c_1 = c_2$ . Using the verification equation (1), we have:

$$s_1^e (H(m||c^e)c)^{\tau(a_1)} = 1 \pmod n \quad \text{and} \quad s_2^e (H(m||c^e)c)^{\tau(a_2)} = 1 \pmod n$$

We further consider what happens if  $\tau(a_1)$  and  $\tau(a_2)$  are relatively prime; as explained in details below, such an assumption is realistic. Using the Bezout theorem and the extended Euclid's algorithm, we can efficiently compute integers  $k$  and  $\ell$  such that

$$k \times \tau(a_1) + \ell \times \tau(a_2) = 1 \tag{2}$$

Then, we can use the so-called Shamir's trick [7] and combine the two signatures:

$$\left( s_1^e (H(m||c^e)c)^{\tau(a_1)} \right)^k \times \left( s_2^e (H(m||c^e)c)^{\tau(a_2)} \right)^\ell = 1^k \times 1^\ell = 1 \pmod n$$

This equation can be rearranged in the following way:

$$(s_1^k \times s_2^\ell)^e \times (H(m||c^e)c)^{k\tau(a_1) + \ell\tau(a_2)} = 1 \pmod n$$

and finally, using the Bezout equation (2) and the notation  $s = s_1^k \times s_2^\ell$ ,

$$s^e (H(m||c^e)c) = 1 \pmod n \tag{3}$$

The *pseudo*-signature  $(s, c)$  can be interpreted as a signature of message  $m$  for a string  $\bar{a}$  such that  $\tau(\bar{a}) = 1$  even if, of course, we are not able to exhibit such a value  $\bar{a}$ . However, *pseudo*-signatures are very useful for an attacker since

they can be immediately converted into valid signature  $(\tilde{s}, \tilde{c}, \tilde{a})$  for **any string**  $\tilde{a}$  by selecting  $\tilde{s} = s^{\tau(\tilde{a})} \bmod n$  and  $\tilde{c} = c$ . Indeed, from equation (3), we have:

$$\begin{aligned} \tilde{s}^e ((H(m||\tilde{c}^e)\tilde{c})^{\tau(\tilde{a})}) &= s^{\tau(\tilde{a}) \times e} ((H(m||c^e)c)^{\tau(\tilde{a})}) \\ &= (s^e (H(m||c^e)c))^{\tau(\tilde{a})} \\ &= 1 \bmod n \end{aligned}$$

Consequently, in the e-cash scenario, if the attacker can withdraw two banknotes of value \$1 and \$2 with the same secret message  $m$ , and then compute the *pseudo*-signature  $(s, c)$ , he can choose a string  $\tilde{a}$  that he would have used to withdraw a \$100 bill and derive from the *pseudo*-signature a valid signature for a \$100 banknote that has never been really withdrawn !

Those observations show that the goal of an attacker is reduced to obtaining two valid signatures  $(s_1, c, a_1)$  and  $(s_2, c, a_2)$  sharing the same  $c$  element and for the same message  $m$ . We now describe how this can be done in a concurrent scenario where two users can simultaneously interact with the signer.

## 4.2 Description of the attack

Let us assume that two users can simultaneously request a partially blind signature, for two different strings  $a_1$  and  $a_2$  respectively, from the same signer. For example, the first user withdraws a \$1 banknote and a second one asks the bank for a \$2 bill. In practice, such a scenario seems realistic if the bank wants to be able to issue e-cash efficiently. Furthermore, the formal security model of Abe and Okamoto [2] explicitly allows *concurrent and interleaving* executions of signature protocols. In the sequel we consider those to users as a single attacker that performs simultaneously two blind signature protocols with the bank.

We further assume that the strings  $a_1$  and  $a_2$  are such that  $\gcd(\tau(a_1), \tau(a_2)) = 1$ . We think that in practice the attacker can easily choose  $a_1$  and  $a_2$  which fulfill this property. However, even if the attacker cannot choose the strings, we know from a well-known theorem of Dirichlet that the probability for two  $k$ -bit integers to be relatively prime tends towards  $6/\pi^2 \approx 0.6$  for large enough values of  $k$ . Consequently, the probability of success of an attacker is larger than one half in any case. Furthermore, this condition may be relaxed since the attack can be easily modified to apply even if  $\gcd(a_1, a_2) = \delta$ . In that case, a pseudo-signature can be converted into a valid signature for any string  $\tilde{a}$  such that  $\delta$  divides  $\tau(\tilde{a})$ . Since the attacker chooses  $\tilde{a}$ , he can for example select an expiration date such that this property is verified.

As explained in section 4.1, since  $\tau(a_1)$  and  $\tau(a_2)$  are assumed to be relatively prime, we can apply the Bezout theorem and efficiently compute, using the extended Euclid's algorithm, two integer  $k$  and  $\ell$  such that equation 2 is verified.

Then, the attack proceeds as described in figure 2; the attacker asks simultaneously for two blind signatures of the same (blinded) message  $m$  for the two different public strings  $a_1$  and  $a_2$ . For each communication, represented by an arrow, we note with an index on the right which protocol it is part of.

Attacker		Signer
Choose a string $a_1$ and $a_2$		
s.t. $\gcd(\tau(a_1), \tau(a_2)) = 1$	$\xrightarrow{a_1} 1$	check the validity of $a_1$
	$\xrightarrow{a_2} 2$	check the validity of $a_2$
		$x_1 \in_U \mathbb{Z}_n^*$
	$\xleftarrow{y_1} 1$	$y_1 = x_1^e \bmod n$
		$x_2 \in_U \mathbb{Z}_n^*$
	$\xleftarrow{y_2} 2$	$y_2 = x_2^e \bmod n$
$r_1 \in_U \mathbb{Z}_n^*, r_2 \in_U \mathbb{Z}_n^*, u \in_U \mathbb{Z}_n^*$		
Compute $k$ and $\ell$ s.t.		
$k\tau(a_1) + \ell\tau(a_2) = 1$		
$\xi = u^e y_1^{k\tau(a_1)} y_2^{\ell\tau(a_2)} \bmod n$		
$\alpha_1 = r_1^e uH(m  \xi) \bmod n$	$\xrightarrow{\alpha_1} 1$	
$\alpha_2 = r_2^e uH(m  \xi) \bmod n$	$\xrightarrow{\alpha_2} 2$	
	$\xleftarrow{t_1, x_1} 1$	$t_1 = (\alpha_1 x_1)^{-d\tau(a_1)} \bmod n$
	$\xleftarrow{t_2, x_2} 2$	$t_2 = (\alpha_2 x_2)^{-d\tau(a_2)} \bmod n$
$c = u x_1^{k\tau(a_1)} x_2^{\ell\tau(a_2)} \bmod n$		
$s = \left( r_1^{\tau(a_1)} t_1 \right)^k \left( r_2^{\tau(a_2)} t_2 \right)^\ell \bmod n$		
$(s, c)$ is a <i>pseudo</i> -signature of the message $m$		

**Fig. 2.** Concurrent attack of the Cao, Lin and Xue protocol [3]

The attacker uses the bezout coefficients to combine  $y_1$  and  $y_2$  into  $\xi$ . At the end of the two interleaved protocol's executions he obtains a *pseudo*-signature  $(s, c)$  for the chosen message  $m$ . Indeed,  $s^e(H(m||c^e)c) = 1 \bmod n$ . As explained in section 4.1, it enables to compute valid signatures for **any** string  $\tilde{a}$ .

## References

1. M. Abe and E. Fujisaki. How to Date Blind Signatures. In *Asiacrypt '96*, LNCS 1163, pages 244–251. Springer-Verlag, 1996.
2. M. Abe and T. Okamoto. Provably Secure Partially Blind Signatures. In *Crypto 2000*, LNCS 1880, pages 271–286. Springer-Verlag, 2000.
3. T. Cao, D. Lin, and R. Xue. A randomized RSA-based partially blind signature scheme for electronic cash. *Computers and Security*, 24(1):44–49, february 2005.
4. D. Chaum. Blind Signatures for Untraceable Payments. In *Crypto '82*, pages 199–203. Plenum, NY, 1983.
5. A. Juels, M. Luby, and R. Ostrovsky. Security of Blind Digital Signatures. In *Crypto '97*, LNCS. Springer-Verlag, 1997.
6. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
7. A. Shamir. On the Generation of Cryptographically Strong Pseudo-Random Sequences. *ACM Transaction on Computer Systems*, 1(1):38–44, February 1983.
8. Y. Tsionis. *Efficient Electronic Cash: New Notions and Techniques*. PhD thesis, Northeastern University, june 1997.