

# A Modification of Jarecki and Saxena Proactive RSA Signature Scheme

Ventzislav Nikov  
Innovation and Development Center Leuven,  
NXP Semiconductors, Belgium,  
venci.nikov@gmail.com

Svetla Nikova  
ESAT-COSIC,  
Katholieke Universiteit Leuven, Belgium,  
svetla.nikova@esat.kuleuven.be

**Abstract**—Luo and Lu proposed *URSA proactive signature scheme*, the core of which is based on a new threshold signature protocol - the so-called *t*-bounded offsetting algorithm. Jarecki *et al.* have shown that the *t*-bounded offsetting algorithm leaks information for the shared secret which can be extended to a key-recovery attack on the URSA proactive signature scheme. Jarecki and Saxena proposed a fix to the scheme of Luo and Lu, turning it to a provably secure proactive RSA signature scheme. The authors also posed two open questions on the proactive RSA signature schemes. In this paper we give a solution to the second open problem posed by Jarecki and Saxena. Namely, we propose a proactive RSA signature scheme which does not require all participants to be active in the signature generation protocol.

## I. INTRODUCTION

A threshold  $(t, n)$  signature scheme enables any group of  $t + 1$  participants to sign a message together. Such a scheme is based on secret sharing of the (private) signature key. It is said that the scheme is *t*-secure if any coalition of up to  $t$  participants cannot forge a signature, and it is said that the scheme is *t*-robust if even in the presence of  $t$  corrupt participants the remaining uncorrupt ones can still efficiently sign any message. We are focusing only on RSA threshold signatures, disregarding the discrete log based schemes.

We assume a set  $\mathcal{P}$  of  $n$  participants  $P_1, \dots, P_n$  who can be modelled by polynomial-time randomized Turing machines. Between any two of them there are private channels, and all of them have access to a dedicated broadcast channel. As usual we consider *active*, *adaptive* and *rushing* adversary who may corrupt some of the participants of the scheme. Proactive security refers to security and availability in the presence of a so-called *mobile adversary*. This is an adversary which is allowed to corrupt new participants during the execution of the protocol with the limitation that it can only control a subset of  $t$  participants during any given period [3].

The URSA protocol that Luo and Lu have proposed relies on a new proactive RSA signature scheme, which allows members in an ad hoc group to make access control decisions in a distributed manner. The most efficient currently known provably secure proactive RSA signature schemes [2], [9] require all members to be involved in the signature generation, i.e. all shares should be available. This requirement is posed as the second open problem for proactive RSA threshold signature schemes in [5]. The URSA proactive signature scheme [7] tries to provide a solution by avoiding this requirement, but

without reducing the efficiency. Recently it has been shown in [6] that this scheme is not secure, and a fix has been proposed in [5]. The new scheme in [5] has the same efficiency as the scheme in [9], but unfortunately also has the same requirement to the signing participants.

Our contribution is a modification of the scheme in [5], which provides a solution to the posed open problem. The rest of this paper is organized as follows: Sect. II describes the URSA proactive signature scheme and the attack against it is presented. Next, in Sect. III we explain Jarecki and Saxena's proactive signature scheme. Last, we propose a modification to the recent schemes in Sect. IV.

## II. THE URSA PROACTIVE SIGNATURE SCHEME

First we provide a description of the proactive *URSA signature scheme* [7] used in the URSA ad hoc network access control protocol. In [6] the authors refer to this scheme as an URSA proactive signature scheme, we will provide a description of the scheme of [7] as given in [6].

A *trusted dealer*  $\mathcal{D}$  generates the standard RSA private/public key pair, i.e. picks two random primes and sets  $N$  to be their product. Thus  $(e, N)$  is the public key, where  $\gcd(e, N) = 1$ , and the private key is  $d$  such that  $ed = 1 \pmod{\phi(N)}$  and  $d < N$ . Then using the Shamir's secret sharing scheme  $\mathcal{D}$  shares the RSA private key  $d$ . To be precise,  $\mathcal{D}$  selects a random polynomial  $f(x)$  of degree  $t$  over  $\mathbb{Z}_N$  subject to  $f(0) = d \pmod{N}$ . Next, every participant of the scheme  $P_i$  (for  $i = 1, \dots, n$ ) receives his share  $ss_i = f(i) \pmod{N}$ . **Remark.** In fact this is a minor modification of Shamir's scheme which is over prime modulus, while the used scheme is over composite modulus.

### A. The Threshold Signature Protocol

As usual the goal of this protocol is to sign a message  $m$  in distributed manner, i.e. to compute  $m^d \pmod{N}$ . The described protocol consists of two phases. First each participant creates its *partial signature* and, second, the signature is reconstructed from a set of at least  $t + 1$  partial signatures.

Denote by  $G$  the group of at least  $t + 1$  participants which are involved in the signature protocol. Note that in [7], [6] the group  $G$  has always  $t + 1$  participants. Since the group can reconstruct the shared secret (i.e. the private key) we have  $d = \sum_{j \in G} ss_j \ell_j^{(G)} \pmod{N}$ , for certain  $\ell_j^{(G)} \in \mathbb{Z}_N$ . Each member

$P_i$  of the group  $G$  computes his partial signature  $s_i^{(G)}$  on  $m$  as  $s_i^{(G)} = m^{d_i^{(G)}} \bmod N$ , where  $d_i^{(G)} = ss_i \ell_i^{(G)} \bmod N$ .

To reconstruct the signature the authors of [7] propose a new, so-called  $t$ -bounded offsetting algorithm. In fact, from now on we will refer to it as  $(t+1)$ -bounded offsetting algorithm since the signing group  $G$  has  $t+1$  members. Notice that

$$d = \sum_{j \in G} d_j^{(G)} - \alpha^{(G)} N \quad (1)$$

(over the integers) for some integer  $\alpha^{(G)} \in [0, |G| - 1]$ , since  $d = \sum_{j \in G} d_j^{(G)} \bmod N$  and  $0 \leq d_j^{(G)} \leq N - 1$  for all  $j \in G$ . But from equation (1) we immediately obtain that the signature is  $s = m^d = (\prod_{j \in G} s_j^{(G)}) m^{-\alpha^{(G)} N} \bmod N$  for some integer  $\alpha^{(G)} \in [0, |G| - 1]$ . Since there are  $|G|$  possible values of  $\alpha^{(G)}$ , namely  $\alpha = 0, 1, \dots, |G| - 1$ , one can try each of the possible values  $Y_\alpha = (\prod_{j \in G} s_j^{(G)}) (m^{-N})^\alpha \bmod N$  and return  $s = Y_\alpha$  if  $Y_\alpha^e = m \bmod N$ . The computational cost of signature algorithm is about one full exponentiation  $m^N$  modulo  $N$ .

**Remark.** The algorithm proposed by Luo and Lu [7] was not proven to be secure. As it was pointed out in [6] the above  $(t+1)$ -bounded offsetting algorithm reveals the value  $\alpha^{(G)}$  which leaks some information about the shared secret, (the RSA private key  $d$ ) to an adversary who corrupts a group  $B$  of  $t$  out of the  $t+1$  participants in  $G$ . Moreover the authors of [6] showed that this information leakage leads to a key-recovery attack against the scheme in [7]. The URSA proactive update protocol is a modification of the classic update protocol [3], [4]. It consists of two phases and relies on the existence of an “update group”  $\Omega$  (of at least  $t+1$  participants). In the first phase the classic update protocol is applied but only the participants from  $\Omega$  are involved. In the second phase all remaining participants get their shares from the members of the update group  $\Omega$ .

### B. An Attack to the URSA Proactive Scheme

We provide here a description of a key-recovery attack from [6] on the proactive RSA signature scheme [7]. As stated in [6] “the roots of this attack lie in the  $(t+1)$ -bounded offsetting algorithm which is the core of the URSA threshold signature protocol.”

For the sake of simplicity let us denote by  $B$  the group of corrupt participants (one always can consider the worst case, i.e.  $|B| = t$ );  $G$  the group of signing participants (in general  $|G| \geq t+1$ , but in [6], [7] the authors consider always  $|G| = t+1$ ); the update group  $\Omega$  (again in general  $|\Omega| \geq t+1$ , but in [6], [7] the authors consider always  $|\Omega| = t+1$ ). It is assumed in [6] that  $B \subset G$  and  $B \subset \Omega$ . Denote the participant  $P_p \in (G \setminus B)$ , note that  $P_p$  can be any uncorrupt participant (i.e. not in  $B$ ) and thus  $G$  depends implicitly on his choice. Recall that by equation (1) we have (over the integers)

$$d = \sum_{j \in B} d_j^{(G)} + d_p^{(G)} - \alpha^{(G)} N. \quad (2)$$

Denote  $S^{(G)} = \sum_{j \in B} d_j^{(G)}$  (over the integers) and  $D^{(G)} = S^{(G)} \bmod N$ . We stress here that  $S^{(G)}$  and  $D^{(G)}$  are values

known by the adversary. In addition to employing the  $(t+1)$ -bounded offsetting algorithm the adversary learns the value  $\alpha^{(G)}$  corresponding to the signing group  $G$ . Now as stated in the Remark from Section II this leaks information about the secret. Namely, it reveals to the adversary whether the shared secret  $d$  is less than or greater than  $D^{(G)}$ .

**Lemma 2.1:** [6] For the defined above notions  $S^{(G)}, D^{(G)}, \alpha^{(G)}, N$  and  $d$  from equation (2) it follows that  $S^{(G)} < \alpha^{(G)} N$  if and only if  $d < D^{(G)}$ , or equivalently that  $S^{(G)} \geq \alpha^{(G)} N$  if and only if  $d \geq D^{(G)}$ .

Based on this observation the attack presented in [6] is as follows. The interval  $[0, N - 1]$  is divided into  $t+1$  equal intervals  $\{[0, D_1 - 1], [D_1, D_2 - 1], \dots, [D_t, N - 1]\}$ , where  $D_1, D_2, \dots, D_t$  are the values  $D^{(G)}$  for  $t$  different choices of the participant  $P_p$ . Denote the participant corresponding to the value  $D_i$  by  $P_{p_i}$  and the corresponding signing group by  $G_i$ . Recall that  $n > 2t$  and that  $P_p$  can be any of the  $n - t$  uncorrupt participants. Thus the attacker first picks  $t$  out of  $n - t$  possibilities for  $P_p$ .

Because of Lemma 2.1 for any run of the threshold signature protocol, involving participants of group  $G_i$ , the attacker learns whether the shared secret  $d$  lies to the left or to the right of the corresponding value  $D_i$ . Thus as stated in [6] “the attacker learns from  $t$  instances of the threshold signature protocol, for  $p = p_1, p_2, \dots, p_t$ , whether  $d$  lies to the left or to the right of these  $D_p$ ’s. Consequently the attacker shrinks the search interval for the secret  $d$  from  $[0, N - 1]$  to some interval  $[D_{p-1}, D_p - 1]$  which is smaller than the original interval by the factor of  $t+1$ .” The shrinking of the interval for the secret is equivalent to saying that the adversary learns the  $\lg(t+1)$  most significant bits of  $d$ . Hence the adversary repeats this procedures every update interval, until he finally restores the secret  $d$ .

### III. JARECKI AND SAXENA’S PROACTIVE RSA SIGNATURE SCHEME

Nearly at the same time when the paper [6] was published, Jarecki and Saxena [5] proposed a fix to the URSA proactive signature scheme, turning it to *provably secure* scheme: “this partial information leakage can be proven harmless once the polynomial sharing used by [7] is replaced by top-level *additive* sharing with second level polynomial sharing back-up.”

A *trusted dealer*  $\mathcal{D}$  generates the standard RSA private/public key pair  $d$  and  $(e, N)$ . Next he chooses a security parameter  $\tau$  (e.g.  $\tau = 80$  for  $|N| = 1024$ ), the maximal number  $r$  of rounds in the lifetime of the system and sets a *prime*  $q$  such that  $q \geq r2^{|N|+\tau}$ . Then  $\mathcal{D}$  shares the RSA private key  $d$  *additively* modulo the *prime*  $q$ , i.e. each participant  $P_i$  holds a share  $ss_i$  (a random number in  $\mathbb{Z}_q$ ) such that  $d = \sum_{i=1}^n ss_i \bmod q$ . Each of these top-level additive shares are (for back-up) shared using the Pedersen’s verifiable secret sharing (VSS) scheme [8] (similarly to Rabin’s [9] proactive RSA scheme). Namely, for each  $ss_i$   $\mathcal{D}$  selects a random polynomial  $f_i(x)$  of degree  $t$  over  $\mathbb{Z}_q$  subject to  $f_i(0) = ss_i \bmod q$ .

Next, every participant of the scheme  $P_i$  (for  $i = 1, \dots, n$ ) receives his share  $ss_i$  and his  $n$  back-up shares  $ss_i^{(j)} = f_j(i) \bmod q$  ( $j = 1, \dots, n$ ). Pedersen's commitment instance  $(p, q, g, h)$ , i.e. commitments of the form  $g^a h^b \bmod p$  are used to commit to every of the coefficients of the polynomials  $f_i(x)$ . We will not describe the pair polynomials  $f_i'(x)$  and additive shares  $ss_i'$  needed for the Pedersen's VSS/commitment schemes.

It is possible to make this scheme more efficient, e.g., [9], [5] by revealing publicly  $|N|/2$  bits of the private key  $d$ . We will not do this in order to keep the notation simpler.

**Remark.** Two differences can be found if we compare the scheme of Luo and Lu [7] (as described in [6]) and the scheme of Jarecki and Saxena [5]:

- the first scheme is over  $\mathbb{Z}_N$ , while the second one is over  $\mathbb{Z}_q$  for  $q \gg N$ .
- while the first scheme uses simple (one level) polynomial sharing of the private key, the second one has two levels: the top-level is additive and the back-up level is polynomial.

The share update protocol is a straightforward modification of the "classic" share update protocol [3], [4] to the additive sharing [1]. Each participant  $P_i$  additively re-shares his share  $ss_i$ , i.e. he randomly picks values  $ss_{i,j} \in \mathbb{Z}_q$  such that  $ss_i^{(old)} = \sum_{j=1}^n ss_{i,j} \bmod q$ . Then  $P_i$  commits to every  $ss_{i,j}$  and sends  $ss_{i,j}$  to  $P_j$ .

Every participant  $P_j$  can now verify the validity of the received additive sub-shares  $ss_{i,j}$ , by using the  $P_i$ 's commitments and the commitments from the setup protocol. Next,  $P_j$  computes its new additive share as  $ss_j^{(new)} = \sum_{i=1}^n ss_{i,j} \bmod q$  and the corresponding commitment to  $ss_j^{(new)}$ .

It is easy to verify that  $\sum_{j=1}^n ss_j^{(new)} \bmod q = \sum_{j=1}^n \sum_{i=1}^n ss_{i,j} \bmod q = \sum_{i=1}^n ss_i^{(old)} \bmod q = d$ . Finally,  $P_j$  shares his new additive share  $ss_j^{(new)}$  using Pedersen's VSS such that all remaining participants can verify that indeed the new additive share of  $P_j$  is shared.

#### A. The Threshold Signature Protocol

The described signature protocol consists of two phases, first each participant creates its *partial signature* and second the signature is reconstructed from all  $n$  partial signatures.

First, every participant  $P_i$  computes its partial signature  $s_i$ , where  $s_i = m^{ss_i} \bmod N$ . If a participant  $P_j$  fails to provide its partial signature, all remaining participants reconstruct his additive share  $ss_j$  and compute the corresponding partial signature  $s_j = m^{ss_j} \bmod N$ . Next the RSA signature is reconstructed using the  $n$ -bounded offsetting algorithm. If the signature reconstruction fails, the faulty partial signatures are traced by executing (more expensive) zero knowledge protocols. Any faulty partial signature is then reconstructed by the uncorrupt participants (by using the back-up shares).

Since all participants together can reconstruct the shared secret by  $d = \sum_{i \in \mathcal{P}} ss_i \bmod q$  (and for all shares  $0 \leq ss_i \leq q-1$ ), therefore for some integer  $\alpha^{(P)} \in [0, n-1]$  it holds

(over the integers) that

$$d = \sum_{i \in \mathcal{P}} ss_i - \alpha^{(P)} q. \quad (3)$$

Analogously to Section II, from equation (3) we immediately obtain that the signature is  $s = m^d = (\prod_{i \in \mathcal{P}} m^{ss_i}) m^{-\alpha^{(P)} q} \bmod N$ . Since there are  $n$  possible values of  $\alpha^{(P)}$ , namely  $\alpha = 0, 1, \dots, n-1$ , one can try each of the possible values  $Y_\alpha = (\prod_{i \in \mathcal{P}} m^{ss_i}) (m^{-q})^\alpha \bmod N$  and return  $s = Y_\alpha$  if  $Y_\alpha = m \bmod N$ . The computational cost of signature algorithm is about one full exponentiation  $m^q$  modulo  $N$ .

**Remark.** The signature protocol that has been described here is more expensive: because of the exponentiation  $m^q$  instead of  $m^N$  ( $q \gg N$ ) and because of  $n$ -bounded instead of  $(t+1)$ -bounded offsetting algorithm (recall that  $n > 2t$ ) is used. We do not count the zero knowledge protocols since they will be rarely used. Another important difference is that all participants should be involved in the protocol.

#### IV. A MODIFICATION OF THE THRESHOLD SIGNATURE PROTOCOL

The fact, that the proactive RSA schemes are build on top of some form of additive rather than polynomial secret sharing, is posed in [5] as the second *open problem* for the proactive RSA solutions. The problem is that "the additive sharing implies that the shares of all temporarily unavailable participants should be reconstructed by the active ones in the signature generation protocol" [5].

In this section we will modify the signature generation protocol avoiding the reconstruction of the missing additive shares and thus we do not require all participants to be available. Denote by  $G$  the group of at least  $t+2$  participants which are involved in the signature protocol. Since the group can reconstruct any participant's  $P_i$  additive share we have  $ss_i = \sum_{j \in G} ss_j^{(i)} \ell_j^{(G)} \bmod q$ , for certain  $\ell_j^{(G)} \in \mathbb{Z}_q$ , using their back-up shares. Thus each member  $P_i$  of the group  $G$  computes his partial signature  $s_i^{(G)}$  on  $m$  as  $s_i^{(G)} = m^{d_i^{(G)}} \bmod N$ , where  $d_i^{(G)} = ss_i + \ell_i^{(G)} \sum_{j \notin G} ss_j^{(j)} \bmod q$ . It is easy to verify that

$$\begin{aligned} \sum_{i \in G} d_i^{(G)} \bmod q &= \sum_{i \in G} ss_i + \sum_{i \in G} \ell_i^{(G)} \sum_{j \notin G} ss_j^{(j)} \bmod q \\ &= \sum_{i \in G} ss_i + \sum_{j \notin G} \sum_{i \in G} \ell_i^{(G)} ss_i^{(j)} \bmod q \\ &= \sum_{i \in G} ss_i + \sum_{j \notin G} ss_j \bmod q \\ &= \sum_{i \in \mathcal{P}} ss_i \bmod q = d. \end{aligned}$$

Further the RSA signature is reconstructed from the partial signatures again by using the  $|G|$ -bounded offsetting algorithm, i.e. for some integer  $\alpha^{(G)} \in [0, |G|-1]$  it holds (over the integers) that  $d = \sum_{i \in G} d_i^{(G)} - \alpha^{(G)} q$ . Analogously we immediately obtain that the signature is

$$s = m^d = \left( \prod_{i \in G} m^{d_i^{(G)}} \right) m^{-\alpha^{(G)} q} \bmod N.$$

Since there are  $|G|$  possible values of  $\alpha^{(G)}$  one can, for each of the possible values  $\alpha$ , compute  $Y_\alpha = (\prod_{i \in G} m^{d_i^{(G)}})(m^{-q})^\alpha \bmod N$  and return as signature  $s = Y_\alpha$  if  $Y_\alpha^e = m \bmod N$ .

If the signature reconstruction fails, the faulty partial signatures can be traced by executing zero knowledge protocols. More precisely  $P_i$  proves in zero-knowledge (as in [5]) first that there is a pair of additive share  $(ss_i, ss_i')$  and second that there are pairs of back-up share  $(ss_i^{(j)}, ss_i^{(j)'})$  (for  $j \notin G$ ). Notice that in total tracing the faulty partial signatures in our scheme will be more expensive compared to the Jarecki and Saxena's scheme.

The security of the modified scheme follows from the observation that any attack on it leads to an attack on the Jarecki and Saxena's scheme. First, note that the two level sharing [5] doesn't prevent the adversary which can speak last during the update protocol to choose the values it owns for the next period (analogous to [6]). Second, note that when  $G = \mathcal{P}$  the proposed scheme coincides with the original scheme.

Similarly to [5], [6] let  $B \subset G$  be the set of  $t$  corrupt participants. Denote  $S^{(G)} = \sum_{j \in B} d_j^{(G)}$ ,  $Z^{(G)} = \sum_{j \in G \setminus B} d_j^{(G)}$  (both over the integers) and  $D^{(G)} = S^{(G)} \bmod q$ , i.e.  $S^{(G)} = D^{(G)} + \beta^{(G)}q$  where  $\beta^{(G)} \in [0, t-1]$ . Employing the  $|G|$ -bounded offsetting algorithm we have

$$d = S^{(G)} + Z^{(G)} - \alpha^{(G)}q \quad (4)$$

and hence  $Z^{(G)} = (d - D^{(G)}) + (\alpha^{(G)} - \beta^{(G)})q$ . Again  $S^{(G)}$ ,  $D^{(G)}$ ,  $\alpha^{(G)}$  and  $\beta^{(G)}$  are values known by the adversary. Here we can make two observations. First, the equation (4) differs from (2) since  $q$  is replaced by  $N$  and the number of honest (uncorrupt) participants is  $k = |G \setminus B| > 1$ . The second observation is that (4) is the same as in [5] (the original security proof) except that we allow  $G \subseteq \mathcal{P}$ . It can be easily proven (as in [5]) that the equation  $Z^{(G)} = (d - D^{(G)}) + (\alpha^{(G)} - \beta^{(G)})q$  prevents the information leakage shown for equation (2) (i.e. Lemma 2.1). Hence the adversary has the same information as in [5]. Although in the security proof in [5] the adversary is provided with additional information, it

can be shown that the adversarial view of the proposed protocol is indistinguishable (with statistical difference no more than  $2^{-\tau}$ ) from a simulation.

## V. CONCLUSIONS

In this paper we propose a modification of the recent proactive RSA signature scheme [5], which does not require the shares of all temporarily unavailable participants to be reconstructed by the active ones in the signature generation protocol. Thus we have shown a solution to the second open problem for proactive RSA signature scheme posed by Jarecki and Saxena.

## ACKNOWLEDGEMENTS

Svetla Nikova was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

## REFERENCES

- [1] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Adaptive security for threshold cryptosystems", *CRYPTO'99*, LNCS 1666, 1999, pp. 98–115.
- [2] Y. Frankel, P. Gemmel, P. MacKenzie, M. Yung, "Proactive RSA", *CRYPTO'97*, LNCS 1294, 1997, pp. 440–454.
- [3] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage", *CRYPTO'95*, LNCS 963, 1995, pp. 339–352.
- [4] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, "Proactive Public Key and Signature Systems", *ACM'97 – Computer and Communication Security*, 1997, pp. 100–110.
- [5] S. Jarecki, N. Saxena, "Further Simplifications in Proactive RSA Signatures", *TCC'05*, LNCS 3378, 2005, pp. 510–528.
- [6] S. Jarecki, N. Saxena, J. Yi "An Attack on the Proactive RSA Signature Scheme in the URSA Ad Hoc Network Access Control Protocol", *SASN'04*, 2004, pp. 1–9.
- [7] H. Luo, S. Lu, "Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks", *Technical Report TR-200030*, Dept. Comp. Sci. UCLA, 2000. Available online at <http://citeseer.ist.psu.edu/luo00ubiquitous.html>
- [8] T. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing", *CRYPTO'91*, LNCS 547, 1992, pp. 129–140.
- [9] T. Rabin, "A Simplified Approach to Threshold and Proactive RSA", *CRYPTO'98*, LNCS 1462, 1998, pp. 89–104.