On Schnorr's Preprocessing for Digital Signature Schemes*

Peter de Rooij
PTT Research†

Abstract

Schnorr's preprocessing algorithms [6, 7] are designed to speed up the 'random' exponentiation often performed by the prover/signer in identification and signature schemes.

In this paper, an attack on these preprocessing algorithms is presented. For the proposed parameters, the attack requires about 2^{31} steps, and 700 identifications or signatures to retrieve the secret key. Here the underlying scheme may be Schnorr, Brickell-McCurley, ElGamal or DSS.

1 Introduction

At Crypto'89, a discrete log based identification protocol and a related signature scheme especially suited for use on smart cards were proposed by C. Schnorr [6]. One optional feature of both was the possibility of using a preprocessing algorithm that substantially speeds up the prover's calculations. However, use of the proposed preprocessing algorithm enables an attack that retrieves the prover's secret key [5].

An improved preprocessing algorithm, resistant to this attack, was proposed in [7]. In this paper, the attack is adapted to the new preprocessing algorithm. This new attack still is applicable to the old preprocessing, and is substantially faster than the old attack from [5].

The preprocessing algorithms can be used in any protocol where a random power of a fixed base is to be computed. The proposed attack can be adapted to any such protocol that satisfies a (rather mild) assumption. Specifically, it is shown to be applicable to Brickell-McCurley, ElGamal and DSS signatures [1, 2, 3].

2 Description of the Algorithm

2.1 Schnorr's Identification Protocol and Signature Scheme

We briefly describe Schnorr's identification protocol and the related signature scheme. For details, see [7].

The following parameters are public: a large prime p , a prime q that divides $p-1$, a primitive q th root of unity $\alpha \in \mathbb{Z}_p$ and a security parameter t . In [7], it is proposed to

*. Part of the results of this paper were presented at Eurocrypt'91 [5].

†. P.O. Box 421, 2260 AK Leidschendam, the Netherlands. E-mail: P.J.N.deRooij@research.ptt.nl

take p and q in the order of 512 bits and 140 bits respectively, and $t = 72$. Each user $\mathcal{A}$ has a secret key $s_{\mathcal{A}} \in \mathbb{Z}_q^*$ and a corresponding public key $v_{\mathcal{A}} = \alpha^{-s_{\mathcal{A}}} \bmod p$.

The identification protocol. A prover $\mathcal{A}$ that wants to prove her identity to verifier $\mathcal{B}$ first picks a random number $r^* \in \mathbb{Z}_q^*$ and sends the *initial commitment* $x^* = \alpha^{r^*} \bmod p$ to $\mathcal{B}$. Then $\mathcal{B}$ returns a random *challenge* $e \in \{0, \dots, 2^t - 1\}$, to $\mathcal{A}$. Next, $\mathcal{A}$ sends $y = r^* + s_{\mathcal{A}} \cdot e \bmod q$ to $\mathcal{B}$. Finally, $\mathcal{B}$ checks $\mathcal{A}$'s proof of identity by verifying if $x^* = \alpha^y v_{\mathcal{A}}^e \bmod p$.

Below, the index $\mathcal{A}$ is dropped, as no confusion is possible. Furthermore, all calculations are performed modulo q , except where indicated otherwise.

The signature scheme. The identification protocol is extended to a signature scheme by replacing the challenge from the verifier by a t -bit hash value $e = h(x^*, m)$ of the initial commitment x^* and the message m to be signed. The signature consists of this hash value and of y as in the identification protocol.

In the sequel, a pair (y, e) is called a *signature*, even if it is constructed in the course of an instance of the identification protocol.

2.2 The preprocessing algorithm

The purpose of the preprocessing algorithm is the reduction of the computational effort of the prover/signer, which is determined by the random exponentiation $\alpha^{r^*} \bmod p$. The preprocessing simulates this random exponentiation by a pseudorandom one that requires a few multiplications only. This is achieved by taking, instead of a random r^* , a linear combination of several independently and randomly chosen numbers $r_i \in \mathbb{Z}_q^*$ for which $x_i = \alpha^{r_i} \bmod p$ is precomputed.

For this purpose, each user initially stores a collection of k such pairs (r_i, x_i) , $0 \leq i < k$. Here k is a security parameter; the proposed value [6, 7] is $k = 8$. Then, for each signature, the pair (r^*, x^*) is chosen as a combination of the currently stored pairs (r_i, x_i) , while the stored collection is 'rejuvenated' by replacing one of these pairs by a similar combination.

We denote this as follows. At time ν , $\nu \geq k$, the pair (r_ν, x_ν) replaces $(r_{\nu-k}, x_{\nu-k})$; both the pair (r^*, x^*) used in the initial commitment and the signature (y, e) receive an index ν . The preprocessing algorithm from [7] is as follows.

Initialization. Load k pairs (r_i, x_i) as above, $0 \leq i < k$; set $\nu := k$;

1. Pick a random permutation $(a_\nu(0), \dots, a_\nu(k-1))$ of $\{0, \dots, k-1\}$;

$$a_\nu(k) := 0; a_\nu(k+1) := k-1;$$

($a_\nu(\cdot)$ determines the order of use of the stored pairs.)

2. $r_\nu^* := r_{\nu-k} + 2r_{\nu-1} \bmod q$; $r_\nu := \sum_{i=0}^{k+1} 2^i r_{a_\nu(i)+\nu-k} \bmod q$;
 $x_\nu^* := x_{\nu-k} \cdot x_{\nu-1}^2 \bmod p$; $x_\nu := \prod_{i=0}^{k+1} (x_{a_\nu(i)+\nu-k})^{2^i} \bmod p$;

3. Keep (r_ν^*, x_ν^*) ready for the next signature;

Keep (r_i, x_i) , $\nu - k + 1 \leq i \leq \nu$ stored; (i.e., replace $(r_{\nu-k}, x_{\nu-k})$ with (r_ν, x_ν))

4. $\nu := \nu + 1$; goto 1 for the next signature.

(The preprocessing from [6] differs in the form of $a_\nu(\cdot)$ only.) This preprocessing requires $2k + 2$ multiplications modulo p and a storage of k pairs (r_i, x_i) [7].

3 A New Attack

3.1 Dependencies between signatures

This section generalizes the attack from [5] to the new preprocessing. Assume that p, q, α, t and some number of successive signatures are available to the enemy. The following equations hold:

$$r_\nu^* = r_{\nu-k} + 2r_{\nu-1}, \quad \nu = k, k+1, \dots \quad (1)$$

$$r_\nu = \sum_{j=0}^{k+1} 2^j r_{a_\nu(j)+\nu-k}, \quad \nu = k, k+1, \dots \quad (2)$$

$$y_\nu = r_\nu^* + se_\nu, \quad \nu = k, k+1, \dots \quad (3)$$

From (1) and (3) it follows that $r_{j-1} = \frac{1}{2}(y_j - se_j - r_{j-k})$ for all $j \geq k$. Repeated substitution using this equality yields the *linking equation*:

$$r_{j+c(k-1)} = (-2)^{-c} r_j - \sum_{l=1}^c (-2)^{l-c-1} (y_{j+l(k-1)+1} - se_{j+l(k-1)+1}). \quad (4)$$

Given the (y_i, e_i) -pairs that occur in the corresponding linking equation (4), any equation of the same form as (2) can be rewritten to a linear equation in $r_0, \dots, r_{k-2}$ and s . For details we refer to the final paper [4].

So, given n different equations of the same form as (2), say for the sequential numbers $i_j, i_0 < i_1 < \dots < i_{n-1}$, and given the signatures (y_i, e_i) with sequential numbers $i_0 < i \leq i_{n-1} + 1$, a set of equations of the following form can be derived:

$$\left[\begin{array}{c|c} M & \begin{matrix} E'_{i_0} \\ \vdots \\ E'_{i_{n-1}} \end{matrix} \end{array} \right] \cdot \left[\begin{matrix} r_0 \\ \vdots \\ r_{k-2} \\ \hline s \end{matrix} \right] = \left[\begin{matrix} Y'_{i_0} \\ \vdots \\ Y'_{i_{n-1}} \end{matrix} \right]. \quad (5)$$

Here the E'_i and Y'_i are known linear combinations of the e_i 's respectively y_i 's, and M is an $n \times (k-1)$ matrix with entries depending on the i_j and $a_{i_j}(\cdot)$ only, see [4].

In the old preprocessing it is possible that only two $a_i(l)$'s are nonzero. The corresponding row of M then has a support of weight 2. In the attack described in [5], three rows with the same support of weight 2 are collected. Then (5) reduces to a system of three equations in three unknowns, which can be solved with high probability. For details, see [5].

That attack will not work here, as it is guaranteed by the choice of the $a_i(l)$'s that all rows of M have a support of weight $k-1$. This implies that the attack will require a $k \times (k-1)$ matrix M . The work factor then increases to about $(k!)^k$ steps, as there are that many different possible M -matrices. This is infeasible for the proposed $k=8$.

The attack presented here will collect two *identical* rows of M . Given that, the entries of M cancel out by subtraction of the two rows i_0 and i_1 of (5). This yields $s(E'_{i_0} - E'_{i_1}) = Y'_{i_0} - Y'_{i_1}$, which provides s if $E'_{i_0} \neq E'_{i_1}$, which happens with overwhelming

probability. Two rows of M are identical if and only if $i_0 \equiv i_1 \pmod{k-1}$ and if the corresponding permutations $a_{i_0}(\cdot)$ and $a_{i_1}(\cdot)$ are identical, see [4].

Thus, if one possesses the signatures (y_i, e_i) for $i_0 < i \leq i_1 + 1$ and if $i_0 \equiv i_1 \pmod{k-1}$, one has a *candidate* to determine the secret key s .

Given a candidate, one can just assume that the related $a_i(\cdot)$ are equal. Then for each of the $k!$ possible values of these permutations, one obtains an estimate $\bar{s}$ for s . The correctness of an estimate is verified by checking whether $\alpha^{-\bar{s}}$ equals the public key v . This modular exponentiation determines the workload per estimate. After checking all $k!$ estimates for a candidate, one either has obtained s (this happens with probability $1/k!$), or the certainty that the permutations are unequal. This whole process is called the *checking of the candidate*. The result of this checking will be either s or "failure".

3.2 An attack

Suppose the enemy possesses a number of consecutive signatures. Each pair of signatures with the same index modulo $k-1$ provides a candidate. An attack could proceed by checking all those candidates in some order. (An example is given in [4].)

3.3 The Complexity of the Attack

The number of checks $\alpha^{-\bar{s}} \stackrel{?}{=} v$ that determines the workload is $k!$ per candidate, and must we check $k!$ candidates, as the probability of success per candidate is $1/k!$. Therefore, the expected number of steps of the attack is $(k!)^2$. For $k = 8$ this is approximately 2^{31} .

The number of required signatures is $\sqrt{\frac{1}{2}\pi(k-1)k!}$ (by the birthday paradox; the factor $k-1$ in the root stems from the fact that we have $k-1$ simultaneous birthday attacks, one for each index modulo $k-1$. For details, see [4]). For $k = 8$, this amounts to 665 consecutive signatures.

3.4 Generalization

Other preprocessing algorithms. The attack will work for any preprocessing where Equations (1) and (3) hold, using $\sim \sqrt{\frac{1}{2}\pi(k-1)N}$ signatures and with complexity N^2 steps, if the analogue of Equation (2) allows N possibilities. For example, the preprocessing from [6] with the proposed parameters has $N = 2^{12}$, so the number of required signatures is about 212; the complexity is 2^{24} . (Compare this to the 2000 signatures and workload of 2^{38} from [5].)

Relaxing the assumptions. The requirement that one collects consecutive signatures by the same user can be dropped. This goes at the expense of increasing the number of required signatures to $k!$ sets of k consecutive ones; the workload remains the same.

Adaption to other signature schemes. The preprocessing can be applied in any situation where random powers of fixed bases are to be computed. This will occur in most discrete log based identification protocols and signature schemes. In general, Equation (3) will in then be different. However, if it is linear in r^* (i.e., its coefficient is known), it is possible to adapt the attack. This will almost always hold, as r^* will

appear in an exponent in a verification equation; for ElGamal [2], DSS [3] and Brickell-McCurley [1] signatures this is the case. The workload is essentially the same in all three cases.

4 Conclusions

The idea of preprocessing the computation of a random power of a fixed base is interesting, as it reduces the effort to only a few multiplications. Especially for smart cards this is worth while. One must take care, however, that not too much information leaks: the replacement of a 'true random number' by a pseudorandom number always provides side information. The preprocessing algorithm as proposed in [7] is not sufficiently secure, as the attack described above shows.

The attack does not seem to depend very heavily on the underlying signature scheme, as it only requires a linear equation in the random exponent. A Schnorr, Brickell-McCurley, ElGamal or DSS signature provides such an equation, as it must hold for a successful verification.

Acknowledgements

The author would like to thank Johan van Tilburg, Arjen Lenstra and Jean-Paul Boly for their many useful comments and Yacov Yacobi for pointing out the issue of adapting the attack to other signature schemes.

References

- [1] E. F. Brickell and K. S. McCurley, "An interactive identification scheme based on discrete logarithms and factoring", *Journal of Cryptology* 5 (1992), no. 1, pp. 29-39.
- [2] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Transactions on Information Theory* IT-31 (1985), no. 4, pp. 469-472.
- [3] National Institute of Technology and Standards, *Specifications for the Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication XX, US. Department of Commerce, February 1 1993.
- [4] P. de Rooij, "On Schnorr's preprocessing for digital signature schemes", *Journal of Cryptology*, to appear.
- [5] ———, "On the security of the Schnorr scheme using preprocessing", *Advances in Cryptology - Proceedings of Eurocrypt'91* (D. W. Davies, ed.), Lecture Notes in Computer Science, vol. 547, Springer-Verlag, 1991, pp. 71-80.
- [6] C. P. Schnorr, "Efficient identification and signatures for smart cards", *Advances in Cryptology - Proceedings of Crypto'89* (G. Brassard, ed.), Lecture Notes in Computer Science, vol. 435, Springer-Verlag, 1990, pp. 239-251.
- [7] ———, "Efficient signature generation by smart cards", *Journal of Cryptology* 4 (1991), no. 3, pp. 161-174.