Fast Generation of Pairs $(k, [k]P)$ for Koblitz Elliptic Curves

[Published in S. Vaudenay and A.M. Youssef, Eds., *Selected Areas in Cryptography*, vol. 2259 of *Lecture Notes in Computer Science*, pp. 151–174, Springer-Verlag, 2001.]

Jean-Sébastien Coron[1], David M'Raïhi[2], and Christophe Tymen[1]

[1] École Normale Supérieure
45 rue d'Ulm, 75230 Paris, France
christophe.tymen@gemplus.com,coron@clipper.ens.fr
[2] Gemplus Card International
3 Lagoon Drive - Suite 300
Redwood City, CA 94065-1566, USA
david.mraihi@gemplus.com

Abstract. We propose a method for increasing the speed of scalar multiplication on binary anomalous (Koblitz) elliptic curves. By introducing a generator which produces random pairs $(k, [k]P)$ of special shape, we exhibit a specific setting where the number of elliptic curve operations is reduced by 25% to 50% compared with the general case when k is chosen uniformly. This generator can be used when an ephemeral pair $(k, [k]P)$ is needed by a cryptographic algorithm, and especially for Elliptic Curve Diffie-Hellman key exchange, ECDSA signature and El-Gamal encryption. The presented algorithm combines normal and polynomial basis operations to achieve optimal performance. We prove that a probabilistic signature scheme using our generator remains secure against chosen message attacks.

Keywords. Elliptic curve, binary anomalous curve, scalar multiplication, accelerated signature schemes, pseudo-random generators.

1 Introduction

The use of the elliptic curves (EC) in cryptography was first proposed by Miller [8] and Koblitz [4] in 1985. Elliptic curves provide a group structure, which can be used to translate existing discrete logarithm-based cryptosystems. The discrete logarithm problem in a cyclic group G of order n with generator g refers to the problem of finding x given some element $y = g^x$ of G. The discrete logarithm problem over an EC seems to be much harder than in other groups such as the multiplicative group of a finite field, and no subexponential-time algorithm is known for the discrete logarithm problem in the class of *non-supersingular*

EC which trace is different from zero and one. Consequently, keys can be much smaller in the EC context, typically about 160 bits.

Koblitz described in [5] a family of elliptic curves featuring several attractive implementation properties. In particular, these curves allow very fast scalar multiplication, *i.e.* fast computation of $[k]P$ from any point P belonging to the curve. The original algorithm proposed by Koblitz introduced an expansion method based on the Frobenius map to multiply points on elliptic curves defined over $\mathbb{F}_2$, $\mathbb{F}_4$, $\mathbb{F}_8$ and $\mathbb{F}_{16}$. An improvement due to Meier and Staffelbach was proposed in [6] and later on, Solinas introduced in [19] an even faster algorithm.

Many EC cryptographic protocols such as the Elliptic Curve Diffie-Hellman for key exchange [13], and the ECDSA for signature [13] require the production of fresh pairs $(k, [k]P)$ consisting of a random integer k and the point $[k]P$. A straightforward way of producing such pairs is to first generate k at random and then compute $[k]P$ using an efficient scalar multiplication algorithm. Another possiblity, introduced and analysed in [16, 18, 17, 14, 15], consists in randomly generating k and $[k]P$ at the same time, so that fewer elliptic curve operations are performed.

In this paper we focus on Koblitz (or anomalous) elliptic curves in $\mathbb{F}_{2^n}$. By introducing a generator producing random pairs $(k, [k]P)$, we are able to exhibit a specific setting where the number of elliptic curve additions is significantly reduced compared to the general case when k is chosen uniformly. The new algorithm combines normal and polynomial basis operations to achieve optimal performance. We provide a security proof for probabilistic signature schemes based on this generator.

The paper is organized as follows: in section 2 we briefly recall the basic definitions of elliptic curves and operations over a finite field of characteristic two. In section 3 we recall the definition of binary anomalous (Koblitz) curves for which faster scalar multiplication algorithms are available. We also recall the specific exponentiation techniques used on this type of curves. In section 4 we introduce the new generator of pairs $(k, [k]P)$. Section 6 provides a security proof for $(k, [k]P)$-based probabilistic signature schemes, through a fine-grained analysis of the distribution of probability of the generator (theorem 2), and using a new result on the security of probabilistic signature schemes (theorem 1). Finally, we propose in section 7 a choice of parameters resulting in a significant increase of speed compared to existing algorithms, with a proven security level.

2 Elliptic Curves on $\mathbb{F}_{2^n}$

2.1 Definition of an elliptic curve

An elliptic curve is the set of points (x, y) which are solutions of a bivariate cubic equation over a field K [7]. An equation of the form:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ , \tag{1}$$

where $a_i \in K$, defines an elliptic curve over K.

In the field $\mathbb{F}_{2^n}$ of characteristic 2, equation (1) can be reduced to the form:

$$y^2 + xy = x^3 + ax^2 + b \text{ with } a, b \in \mathbb{F}_{2^n} \ .$$

The set of points on an elliptic curve, together with a special point $\mathcal{O}$ called the *point at infinity*, has an abelian group structure and therefore an addition operation. The formula for this addition is provided in [13].

2.2 Computing a multiple of a point

The operation of adding a point P to itself d times is called *scalar multiplication* by d and denoted $[d]P$. Scalar multiplication is the basic operation for EC protocols. Scalar multiplication in the group of points of an elliptic curve is analog to the exponentiation in the multiplicative group of integers modulo a fixed integer p.

Computing $[d]P$ is usually done with the *addition-subtraction method* based on the *nonadjacent form* (NAF) of the integer d, which is a signed binary expansion without two consecutive nonzero coefficients:

$$d = \sum_{i=0}^{\ell-1} c_i 2^i \ ,$$

with $c_i \in \{-1, 0, 1\}$ and $c_i \cdot c_{i+1} = 0$ for all $i \geq 0$. The NAF is said to be optimal because each positive integer has a unique NAF, and the NAF of d has the fewest nonzero coefficients of any signed binary expansion of d [2]. An algorithm for generating the NAF of any integer in described in [9].

3 Anomalous Binary Curves

3.1 Definition and Frobenius map

The *anomalous binary curves* or Koblitz curves [5] are two curves E_0 and E_1 defined over $\mathbb{F}_2$ by

$$E_a : y^2 + xy = x^3 + ax^2 + 1 \text{ with } a \in \{0, 1\} \ . \tag{2}$$

We define $E_a(\mathbb{F}_{2^n})$ as the set of points (x, y) which are solutions of (2) over $\mathbb{F}_{2^n}$.

Since the anomalous curves are defined over $\mathbb{F}_2$, if $P = (x, y)$ is in $E_a(\mathbb{F}_{2^n})$, then the point (x^2, y^2) is also in $E_a(\mathbb{F}_{2^n})$. In addition, it can be checked that:

$$(x^4, y^4) + 2(x, y) = (-1)^{1-a}(x^2, y^2) \ , \tag{3}$$

where $+$ holds for the addition of points in the curve. Let τ be the Frobenius map over $\mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$

$$\tau(x, y) = (x^2, y^2) \ .$$

Equation (3) can be rewritten for all $P \in E_a(\mathbb{F}_{2^n})$ as

$$\tau^2 P + [2]P = (-1)^{1-a}\tau P \ .$$

This shows that the squaring map is equivalent to a multiplication by the complex number τ satisfying

$$\tau^2 + 2 = (-1)^{1-a}\tau \ ,$$

and we say that E_a has a *complex multiplication* by τ [5]. Consequently, a point on E_a can be multiplied by any element of the ring $\mathbb{Z}[\tau] = \{x + y \cdot \tau \,|\, x, y \in \mathbb{Z}\}$.

3.2 Faster scalar multiplication

The advantage of using the multiplication by τ is that squaring is very fast in $\mathbb{F}_{2^n}$. Consequently, it is advantageous to rewrite the exponent d as a signed τ-adic NAF

$$d = \sum_{i=0}^{n+1} e_i \tau^i \mod (\tau^n - 1) \ ,$$

with $e_i \in \{-1, 0, 1\}$ and $e_i \cdot e_{i+1} = 0$. This representation is based on the fact that $\mathbb{Z}[\tau]$ is an euclidian ring. An algorithm for computing the τ-adic NAF is given in [19]. This encoding yields the following scalar multiplication algorithm:

Algorithm 1 : Addition-substraction method with τ-adic NAF

Input:P
Output:Q
$Q \leftarrow [e_{n+1}]P$
for $i \leftarrow n$ to 0 do
    $Q \leftarrow \tau Q$
    if $e_i = 1$ then $Q \leftarrow Q + P$
    if $e_i = -1$ then $Q \leftarrow Q - P$
return Q

The algorithm requires approximately $n/3$ point additions instead of n doubles and $n/3$ additions for the general case [19]. If we neglect the cost of squarings, this is four times faster.

As in the general case, it is possible to reduce the number of point additions by precomputing and storing some "small" τ-adic multiples of P. [19] describes an algorithm which requires the storage of

$$C(\omega) = \frac{2^\omega - (-1)^\omega}{3} \text{ points} \ ,$$

where ω is a trade-off parameter. Precomputation requires $C(\omega) - 1$ elliptic additions, and the scalar multiplication itself requires approximately

$$\frac{n}{\omega + 1} \text{ elliptic additions} \ ,$$

which gives a total workload of

$$\simeq \frac{2^\omega}{3} + \frac{n}{\omega+1} \text{ elliptic additions } .$$

For example, for the 163-bit curve $E_1(\mathbb{F}_{2^{163}})$ and $\omega = 4$, a scalar multiplication can be performed in approximately 35 additions, instead of 52 without precomputation.

When P is known in advance, as is the case for protocols such as Elliptic Curve Diffie-Hellman or ECDSA, it is possible to precompute and store the "small" τ-adic multiples of P once for all. The real time computation that remains is the scalar multiplication itself, which requires around $n/(\omega+1)$ operations when $C(w)$ points are stored. For example, for the 163-bit curve $E_1(\mathbb{F}_{2^{163}})$, a scalar multiplication can be performed with $\omega = 7$ in about 19 additions if 43 points are stored.

In the next section we describe an algorithm for producing random pairs $(k, [k]P)$ which requires even fewer additions for approximately the same number of points stored in memory. This algorithm appears to be well-suited for constrained environments such as smart-cards.

4 Fast Generation of $(k, [k]P)$

4.1 A simple generator

Many EC cryptographic protocols such as Elliptic Curve Diffie-Hellman for key exchange [13] and ECDSA for signature [13] require to produce pairs $(k, [k]P)$ consisting of a random integer k in the interval $[0, q-1]$ and the point $[k]P$, where q is a large prime divisor of the order of the curve, and P is a fixed point of order q.

For ECDSA this is the initial step of signature generation. The x coordinate of $[k]P$ is then converted into an integer c modulo q and the signature of m is (c, s) where $s = (H(m) + d \cdot c)/k \mod q$ and d is the private key associated to the public key $Q = d.P$.

[1] describes a simple method for generating random pairs of the form (x, g^x). This method can be easily adapted to the elliptic curve setting for computing pairs $(k, [k]P)$, where P is a point of order q.

Preprocessing:
Generate t integers $k_1, \ldots, k_t \in \mathbb{Z}_q$.
Compute $P_j = k_j.P$ for each j and store the k_j's and the P_j's in a table.
Pair generation:
Randomly generate $S \subset [1, t]$ such that $|S| = \kappa$.
Let $k = \sum_{i \in S} k_j \mod q$.
Let $Q = \sum_{i \in S} P_j$ and return (k, Q).

The algorithm requires $\kappa - 1$ elliptic curve additions. Of course, the generated k is not uniformly distributed and the parameters have to be chosen with great care so that the distribution of the generated k is close to the uniform random distribution.

4.2 The new generator

We consider the generator of figure 1 which produces random pairs of the form $(k, [k]P)$ on a Koblitz curve defined over $\mathbb{F}_{2^n}$.

Preprocessing:

Generate t integers $k_1, \ldots, k_t \in \mathbb{Z}_q$.
Compute $P_j = k_j.P$ for each j
Store the k_j's and the P_j's in a table.

Pair generation:

Generate κ random values $s_i = \pm 1$
Generate κ random integers $e_i \in [0, n-1]$.
Generate κ random indices $r_i \in [1, t]$.
Let $k = \sum_{i=1}^{\kappa} s_i \cdot \tau^{e_i} \cdot k_{r_i} \mod q$.
Let $Q = \sum_{i=1}^{\kappa} s_i \cdot \tau^{e_i} \cdot P_{r_i}$.
Return (k, Q).

Fig. 1. Generation of $(k, [k]P)$ pairs on Koblitz curves

The difference with the previous generator is the use of the Frobenius map τ, which increases the entropy of the generated k. The new generator requires $\kappa - 1$ elliptic curve additions and t points stored in memory. In the next section we describe an efficient implementation of the new generator.

4.3 Implementing the generator

The new generator uses the Frobenius map τ extensively, as on average $\kappa \cdot n/2$ applications of τ are performed for each generated pair, which represents $\kappa \cdot n$ squarings.

Squaring comes essentially for free when $\mathbb{F}_{2^n}$ is represented in terms of a *normal basis*: a basis over $\mathbb{F}_{2^n}$ of the form

$$\{\theta, \theta^2, \theta^{2^2}, \ldots, \theta^{2^{n-1}}\} \ .$$

Namely, in this representation, squaring a field element is accomplished by a one-bit cyclic rotation of the bitstring representing the element.

Elliptic curve additions will be performed using a *polynomial basis* representation of the elements, for which efficient algorithms for field multiplication and inversion are available. A polynomial basis is a basis over $\mathbb{F}_{2^n}$ of the form

$$\{1, x, x^2, \ldots, x^{n-1}\} \ .$$

The points P_j are stored using a normal basis representation. When a new pair is generated, the point $\tau^{e_i} \cdot P_{r_i}$ is computed by successive rotations of the coordinates of P_{r_i}. Then $\tau^{e_i} \cdot P_{r_i}$ is converted into a polynomial basis representation and it is added to the accumulator Q. To convert from normal to polynomial basis, we simply store the change-of-base matrix. The conversion's time is then approximately equivalent to one field multiplication, and this method requires to store $O(n^2)$ bits.

Before a new pair $(k, [k]P)$ is computed, the integers e_i's are sorted: $e_1 \geq e_2 \geq \ldots \geq e_\kappa$, so that k can be rewritten as

$$k = \tau^{e_\kappa} \left(s_\kappa \cdot k_{r_\kappa} + \tau^{e_{\kappa-1}-e_\kappa} \left(s_{\kappa-1} \cdot k_{r_{\kappa-1}} + \ldots \right) \right) \ .$$

The integer k is computed in the ring $\mathbb{Z}[\tau]$ as $k = k' + k'' \cdot \tau$ where $k', k'' \in \mathbb{Z}$. The element $k \in \mathbb{Z}[\tau]$ is finally converted into an integer by replacing τ by an integer T in $\mathbb{Z}_q$ solution of the equation

$$T^2 + 2 = (-1)^{1-a} T \mod q \ ,$$

so that for any point Q, we have $\tau(Q) = [T]Q$.

The implementation of the generator is summarized in figure 2.

5 Lattice Reduction Attacks and Hidden Subsets

When the generator is used in ECDSA, each signature (c, s) of a message m yields a linear equation

$$k \cdot s = H(m) + d \cdot c \mod q \ ,$$

where d is the unknown secret key and k is a sum of the hidden terms $\pm\tau^i \cdot k_i$.

The generator of [1] described in section 4.1 for which k is a sum of the hidden terms k_i has been attacked by Nguyen and Stern [11]. However, the attack requires the number of hidden k_i to be small (around 45 for a 160-bit integer k). The security of the generator relies on the difficulty of the hidden-subset sum problem studied in [11]: given a positive integer M and $b_1, \ldots, b_m \in \mathbb{Z}_M$, find $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_M$ such that each b_i is some subset sum of $\alpha_1, \ldots, \alpha_n$ modulo M.

For the new generator, if one simply considers all $\tau^i \cdot k_i$ to be hidden, this yields a large number of hidden terms ($n \cdot t$, where n is the field size and t the number of stored points) which can not be handled by [11]. We did not find any way of adapting [11] to our new generator.

Preprocessing:

Generate t random integers $k_1, \ldots, k_t \in \mathbb{Z}_q$.
Compute $P_j = k_j.P$ for each j
Store the k_j's and the P_j's in normal basis.

Pair generation:

Generate κ random integers $e_i \in [0, n-1]$
Sort the e_i: $e_1 \geq e_2 \geq \ldots \geq e_\kappa$
Set $e_{\kappa+1} \leftarrow 0$
Set $Q \leftarrow \mathcal{O}$ and $k \leftarrow 0$.
For $i \leftarrow 1$ to κ do:
  Generate a random integer $r \in [1, t]$
  Generate a random $s \leftarrow \pm 1$
  Compute $R \leftarrow s \cdot \tau^{e_i} \cdot P_r$ in normal basis.
  Convert R into polynomial basis.
  Compute $Q \leftarrow Q + R$
  Compute $k \leftarrow \tau^{e_i - e_{i+1}} \cdot (s \cdot k_j + k)$ in $\mathbb{Z}[\tau]$.
Convert k into an integer.
Return (k, Q).

Fig. 2. Algorithm for implementing the generator of $(k, [k]P)$ pairs for Koblitz curves

6 Security Proof for Signature Schemes Using the New Generator

Since the generated integers k are not to be uniformly distributed, the security might be considerably weakened when the generator is used in conjunction with a signature scheme, a key-exchange scheme or an encryption scheme. In this section, we provide a security proof in the case of probabilistic signature schemes.

In the following, we relate the security of a signature scheme using a truly random generator with the security of the same signature scheme using our generator. Resistance against adaptive chosen message attacks is considered. This question has initially been raised by [12], and we improve the result of [12, p. 9].

Let $\mathcal{S}$ be a probabilistic signature scheme. Denote by $\mathcal{R}$ the set of random elements used to generate the signature. In our case of interest, $\mathcal{R}$ will be $\{0, \ldots, q-1\}$. Let G be a random variable on $\mathcal{R}$. Define $\mathcal{S}_G$ as the signature scheme identical to $\mathcal{S}$, except that its generation algorithm uses G as random source instead of a truly random number generator.

The following theorem shows that if a signature scheme using a truly random number generator is secure, the corresponding signature scheme using G will be secure if the distribution of G is sufficiently close to the uniform distribution. The proof is given in appendix.

If X is a random variable on a set Ω, we denote by $\delta_2(X)$ the statistical distance defined by $\delta_2(X) := \left(\sum_{\omega\in\Omega}\left|\Pr(X=\omega)-\frac{1}{|\Omega|}\right|^2\right)^{1/2}$. In the same way, we define $\delta_1(X) := \sum_{\omega\in\Omega}\left|\Pr(X=\omega)-\frac{1}{|\Omega|}\right|$.

Theorem 1. *Let A_G be an adaptive chosen message attack against the signature scheme $\mathcal{S}_G$, during which at most m signature queries are performed. Let A be the corresponding attack on the signature scheme $\mathcal{S}$. The probabilities of existential forgery satisfy*

$$|\Pr(A \textit{ succeeds}) - \Pr(A_G \textit{ succeeds})| \leq$$

$$\frac{(1+|\mathcal{R}|\delta_2(G)^2)^{m/2}-1}{(1+|\mathcal{R}|\delta_2(G)^2)^{1/2}-1}\sqrt{|\mathcal{R}|\Pr(A \textit{ succeeds})}\delta_2(G) \ .$$

Note that asymptotically for $|\mathcal{R}|\delta_2(G)^2 \ll 1$, the bound of theorem 1 yields the inequality

$$|\Pr(A \text{ succeeds}) - \Pr(A_G \text{ succeeds})| \leq m\sqrt{|\mathcal{R}|\Pr(A \text{ succeeds})}\delta_2(G) \ , \quad (4)$$

which has to be compared to the inequality of [12],

$$|\Pr(A \text{ succeeds}) - \Pr(A_G \text{ succeeds})| \leq m\delta_1(G) \ .$$

In the following, we consider our generator of pairs $(k, [k]P)$ of section 4, which we denote by k, and compute its statistical distance $\delta_2(k)$ to the uniform distribution. Using the previous theorem with $G = k$ and $\mathcal{R} = \{0, \ldots, q-1\}$, this will provide a security proof for a signature scheme using our generator.

The following theorem is a direct application of a result exposed in [12]. It gives a bound on the expectation of $\delta_2(k)^2$, this expectation being considered on a uniform choice of $k_1, \ldots, k_t$.

Theorem 2. *If the k_i are independent random variables uniformly distributed in $\{0, \ldots, q-1\}$, then the average of $\delta_2(k)^2$ over the choice of $k_1, \ldots, k_t$ satisfies*

$$E[\delta_2(k)^2] \leq \frac{1}{(2n)^\kappa \binom{t}{\kappa}} \ .$$

In order to use this inequality, we have to link $\delta_2(k)$ to $E[\delta_2(k)^2]$; a simple application of Markov's inequality yields:

Theorem 3. *Let $\epsilon > 0$. With probability at least $1-\epsilon$ (this probability being related to a uniform choice of $k_1, \ldots, k_t$), we have*

$$\delta_2(k) \leq \sqrt{\frac{E[\delta_2(k)^2]}{\epsilon}} \ .$$

Theorem 1 shows that the parameter which measures the security of the signature scheme using our generator is $\sqrt{|\mathcal{R}|}\delta_2(G) = \sqrt{q} \cdot \delta_2(k)$. In table 1 we summarize several values of the bound on $\sqrt{q \cdot E[\delta_2(k)^2]}$ of theorem 2, which using theorem 3 provides an upper bound for $\sqrt{q} \cdot \delta_2(k)$. We stress that the number κ of points to be stored has to be corrected by the amount of data that are required to convert from normal to polynomial basis. Roughly, one must add to κ the equivalent amount of $n/2$ points of the curve, to obtain the total amount of storage needed.

κ/t	25	50	100	150	200
10	31	26	21	18	16
14	15	7	0	-4	-6
16	6	-2	-10	-15	-18
18	-1	-10	-19	-25	-28
20	-9	-19	-29	-35	-39
25	-27	-40	-53	-60	-65

Table 1. $\log_2 \sqrt{q \cdot E[\delta_2(k)^2]}$ for various values of κ and t for $n = 163$

For example, consider the ECDSA signature scheme using our generator with a field size $n = 163$, $\kappa - 1 = 15$ point additions and $t = 100$ precomputed points. Assume that up to $m = 2^{16}$ messages can be signed by the signer. Using table 1, we have $\sqrt{q \cdot E[\delta_2(k)^2]} \approx 2^{-10}$. Using the inequality of theorem 3, we know that, except with probability 2^{-10}, we have $\sqrt{q}\delta_2(k) \leq 2^{-10}/2^{-5} = 2^{-5}$. Assume that for a given time bound, the probability of any attack $\mathcal{A}$ breaking the ECDSA signature scheme with a truly random generator after $m = 2^{13}$ signature queries, is smaller than 2^{-60} for $n = 163$. Then the probability of breaking the ECDSA signature scheme with our generator in the same time bound is smaller than

$$\Pr\left(A_G \text{ succeeds}\right) \leq 2^{13} \cdot \sqrt{2^{-60}} \cdot 2^{-5} = 2^{-19} \ .$$

This shows that the ECDSA signature scheme remains secure against chosen message attacks when using our generator for this set of parameters.

7 Parameters and Performances

We propose two sets of parameters for the field size $n = 163$. The first one is $\kappa = 16$ and $t = 100$ (which corresponds to 15 additions of points), the second is $\kappa = 11$ and $t = 50$ (which corresponds to 10 additions). The first set of parameters provides a provable security level according to the previous section, whereas the second set of parameters lies in a grey area where the existing attacks by lattice reduction do not apply, but security is not proven.

Recall that the scalar multiplication algorithm described in section 3.2 requires 19 elliptic curve additions with 43 points stored. Thus, the two proposed parameter sets induce a 21% and a 47% speed-up factor, respectively[1].

8 Conclusion

We have introduced a new generator of pairs $(k, [k]P)$ for anomalous binary curves. This pairs generator can be used for key exchange (ECDH), signature (ECDSA) and encryption (El-Gamal schemes). We have shown that for an appropriate choice of parameters, a probabilistic signature scheme using our generator remains secure against chosen message attacks. This result can be extended to key exchange schemes and encryption schemes.

We have provided a first set of parameters which provides a speed-up factor of 21% over existing techniques, with a proven security level. The second set of parameters provides a speed-up factor of 47%, but no security proof is available. However, since security is proven for slightly larger parameters, this provides a convincing argument to show that the generator has a sound design and should be secure even for smaller parameters.

9 Acknowledgements

We would like to thank Richard Schroeppel for a careful reading of the preliminary version of this paper, and for many useful comments. We are also grateful to Jacques Stern for his valuable input.

References

1. V. Boyko, M. Peinado, and R. Venkatesan. Speeding up discrete log and factoring based schemes via precomputations. In *Advances in Cryptology - Eurocrypt '98*, pages 221–235. Springer Verlag, 1998.
2. D.M. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, 27:129–146, 1998.
3. B.S. Kaliski Jr. and T.L. Yin. Storage-efficient finite field basis conversion. In *Selected areas in Cryptography - SAC'98*, volume 1556, 1998.
4. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
5. N. Koblitz. CM-curves with good cryptographic properties. In Joan Feigenbaum, editor, *Advances in Cryptology - Crypto '91*, pages 279–287, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science Volume 576.
6. W. Meier and O. Staffelbach. Efficient multiplication on certain non-supersingular elliptic curves. In *Advances in Cryptology - Crypto '92*, volume LNCS 740, pages 333–344. Springer Verlag, 1993.

[1] If we neglect the cost of squaring the P_j's, converting from normal to polynomial basis and computing k.

7. A.J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
8. V.S. Miller. Use of elliptic curves in cryptography. In Springer Verlag, editor, *Proceedings of Crypto 85*, volume LNCS 218, pages 417–426. Springer Verlag, 1986.
9. F. Morain and J. Olivos. Speeding up the computation of an elliptic curve using addition-subtraction chains. *Inform. Theory Appl.*, 24:531–543, 1990.
10. P. Nguyen. *La gomtrie des nombres en cryptologie*. PhD thesis, Universit de Paris 7, 1999.
11. P. Nguyen and J. Stern. The hardness of the hidden subset sum problem and its cryptographic implications. In Michael Wiener, editor, *Advances in Cryptology - Crypto'99*, pages 31–46, Berlin, 1999. Springer-Verlag. Lecture Notes in Computer Science.
12. Phong Nguyen, Igor Shparlinsky, and Jacques Stern. Distribution of Modular Subset Sums and the Security of the Server Aided Exponentiation. In *Workshop on Cryptography and Computational Number Theory*, 1999.
13. IEEE P1363. *Standard Specifications for Public Key Cryptography*. August 1998.
14. P. de Rooij. On the security of the Schnorr scheme using preprocessing. In Donald W. Davies, editor, *Advances in Cryptology - EuroCrypt '91*, pages 71–80, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science Volume 547.
15. P. de Rooij. On Schnorr's preprocessing for digital signature schemes. In Tor Helleseth, editor, *Advances in Cryptology - EuroCrypt '93*, pages 435–439, Berlin, 1993. Springer-Verlag. Lecture Notes in Computer Science Volume 765.
16. P. de Rooij. Efficient exponentiation using precomputation and vector addition chains. In Alfredo De Santis, editor, *Advances in Cryptology - EuroCrypt '94*, pages 389–399, Berlin, 1995. Springer-Verlag. Lecture Notes in Computer Science Volume 950.
17. C. P. Schnorr. Efficient identification and signatures for smart cards. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology - EuroCrypt'89*, pages 688–689, Berlin, 1989. Springer-Verlag. Lecture Notes in Computer Science Volume 434.
18. C. P. Schnorr. Efficient identification and signatures for smart cards. *Journal of Cryptology*, 4:161–174, 1991.
19. J.A. Solinas. An improved algorithm for arithmetic on a family of elliptic curves. In Burt Kaliski, editor, *Advances in Cryptology - Crypto '97*, pages 357–371, Berlin, 1997. Springer-Verlag. Lecture Notes in Computer Science Volume 1294.

A Proof of Theorem 1

Theorem 1. *Let $\mathcal{S}$ be a probabilistic signature scheme. Let $\mathcal{R}$ be the set from which the signature generation algorithm chooses a random element when generating a signature. Let G be a random variable in $\mathcal{R}$, and $\mathcal{S}_G$ the scheme derived from $\mathcal{S}$ which uses G as random source instead of a random oracle for the signature generation. Let A_G be an adaptative attack with m chosen messages on $\mathcal{S}_G$. If A is the corresponding attack on $\mathcal{S}$, then the probabilities of existential forgery satisfy*

$$|\Pr(A \text{ succeeds}) - \Pr(A_G \text{ succeeds})| \leq$$

$$\frac{(1+|\mathcal{R}|\delta_2(G)^2)^{m/2}-1}{(1+|\mathcal{R}|\delta_2(G)^2)^{1/2}-1}\sqrt{|\mathcal{R}|\Pr(A \text{ succeeds})}\delta_2(G) \ .$$

Proof. An adaptative attack with m chosen messages makes m queries to a signature oracle. At each call, this oracle picks a random r in $\mathcal{R}$, and uses this r to produce a signature. If the signature scheme is $\mathcal{S}$, r is chosen uniformly in $\mathcal{R}$, and is thus equal to the value of a random variable U uniformly distributed in $\mathcal{R}$. If the signature scheme is $\mathcal{S}_G$, r is the value of the random variable G. Consequently, an attack with m chosen messages depends on a random variable defined over the probability space $\mathcal{R}^m$. This variable is either $\boldsymbol{U} = (U_1, \ldots, U_m)$ in the case of an attack against $\mathcal{S}$, or $\boldsymbol{G} = (G_1, \ldots, G_m)$ in the case of an attack against $\mathcal{S}_G$, where the U_i are pairwise independent and follow the same distribution as U, and the G_i are pairwise independent and follow the same distribution as G.

The following proof is a refinement of the result that can be found in [12] concerning accelerated signatures schemes. First note that as A and A_G are the same attacks (that is, are the same Turing machines making calls to the same signature oracle except that they use different random sources), for all $\boldsymbol{r} = (r_1, \ldots, r_m) \in \mathcal{R}^m$,

$$\Pr(A \text{ succeeds} | \boldsymbol{U} = \boldsymbol{r}) = \Pr(A_G \text{ succeeds} | \boldsymbol{G} = \boldsymbol{r}) \ .$$

Thus, using Bayes formula, we get

$$\begin{gathered} |\Pr(A \text{ succeeds}) - \Pr(A_G \text{ succeeds})| \leq \\ \sum_{\boldsymbol{r}=(r_1,\ldots,r_m)\in\mathcal{R}^m} |\Pr(\boldsymbol{G} = \boldsymbol{r}) - \Pr(\boldsymbol{U} = \boldsymbol{r})| \Pr(A \text{ succeeds} | \boldsymbol{U} = \boldsymbol{r}) \ . \end{gathered} \tag{5}$$

Using the triangular inequality, the independence of the U_i and of the G_i, and the equidistribution property, we get also that

$$\begin{gathered} |\Pr(\boldsymbol{U} = \boldsymbol{r}) - \Pr(\boldsymbol{G} = \boldsymbol{r})| \leq \\ \sum_{k=1}^{m} \left(\prod_{1\leq i<k} \Pr(G = r_i) \right) |\Pr(U = r_k) - \Pr(G = r_k)| \left(\prod_{m\geq i>k} \Pr(U = r_i) \right) \ , \end{gathered} \tag{6}$$

with the convention that the product of zero terms is equal to 1.

Consequently, if we denote, for $k = 1, \ldots, m$, by $a_k(\boldsymbol{r})$ the quantity

$$\left(\prod_{1\leq i<k} \Pr(G = r_i) \right) |\Pr(U = r_k) - \Pr(G = r_k)| \left(\prod_{m\geq i>k} \Pr(U = r_i) \right) \ ,$$

equation (5) can be rewritten as

$$\begin{gathered} |\Pr(A \text{ succeeds}) - \Pr(A_G \text{ succeeds})| \leq \\ \sum_{k=1}^{m} \sum_{\boldsymbol{r}\in\mathcal{R}^m} a_k(\boldsymbol{r}) \Pr(A \text{ succeeds} | \boldsymbol{U} = \boldsymbol{r}) \ . \end{gathered} \tag{7}$$

Using Cauchy's inequality,

$$\sum_{\boldsymbol{r}\in\mathcal{R}^m} a_k(\boldsymbol{r})\Pr(A \text{ succeeds}|\boldsymbol{U}=\boldsymbol{r}) =$$

$$\sum_{\boldsymbol{r}\in\mathcal{R}^m} \left(|\mathcal{R}|^{m/2}a_k(\boldsymbol{r})\right)\left(|\mathcal{R}|^{-m/2}\Pr(A \text{ succeeds}|\boldsymbol{U}=\boldsymbol{r})\right) \leq \tag{8}$$

$$\left(\sum_{\boldsymbol{r}\in\mathcal{R}^m} |\mathcal{R}|^m a_k(\boldsymbol{r})^2\right)^{1/2}\left(\sum_{\boldsymbol{r}\in\mathcal{R}^m} |\mathcal{R}|^{-m}\Pr(A \text{ succeeds}|\boldsymbol{U}=\boldsymbol{r})^2\right)^{1/2} .$$

And as $\Pr(A \text{ succeeds}|\boldsymbol{U}=\boldsymbol{r}) \leq 1$,

$$\sum_{\boldsymbol{r}\in\mathcal{R}^m} |\mathcal{R}|^{-m}\Pr(A \text{ succeeds}|\boldsymbol{U}=\boldsymbol{r})^2 \leq$$

$$\sum_{\boldsymbol{r}\in\mathcal{R}^m} |\mathcal{R}|^{-m}\Pr(A \text{ succeeds}|\boldsymbol{U}=\boldsymbol{r}) = \Pr\left(A \text{ succeeds}\right) , \tag{9}$$

because U is uniformly distributed over $\mathcal{R}$. Returning to the definition of $a_k(\boldsymbol{r})$, and using once again the uniformity of U, one sees that

$$|\mathcal{R}|^m a_k(\boldsymbol{r})^2 \leq |\mathcal{R}|\left(\prod_{1\leq i<k} |\mathcal{R}|\Pr(G=r_i)^2\right)\ |\Pr(U=r_k)-\Pr(G=r_k)|^2 . \tag{10}$$

Now, one needs to note that

$$\sum_{r_i\in\mathcal{R}} |\mathcal{R}|\Pr(G=r_i)^2 =$$

$$\sum_{r_i\in\mathcal{R}} |\mathcal{R}|\left(\left(\Pr(G=r_i)-\frac{1}{|\mathcal{R}|}\right)^2 - 1/|\mathcal{R}|^2 + (2/|\mathcal{R}|)\Pr(G=r_i)\right) =$$

$$|\mathcal{R}|\delta_2(G)^2 + 1 .$$

Thus, the inequality (10) becomes,

$$\sum_{\boldsymbol{r}\in\mathcal{R}^m} |\mathcal{R}|^m a_k(\boldsymbol{r})^2 \leq$$

$$|\mathcal{R}|\left(1+|\mathcal{R}|\delta_2(G)^2\right)^{k-1}\sum_{r_k\in\mathcal{R}} |\Pr(U=r_k)-\Pr(G=r_k)|^2 =$$

$$|\mathcal{R}|\left(1+|\mathcal{R}|\delta_2(G)^2\right)^{k-1}\delta_2(G)^2 .$$

Returning to inequality (7), and using (8) and (9), we finally get:

$$|\Pr(A \text{ succeeds}) - \Pr(A_G \text{ succeeds})| \leq$$
$$\sum_{k=1}^{m} \left(|\mathcal{R}| \left(1 + |\mathcal{R}|\delta_2(G)^2\right)^{k-1} \delta_2(G)^2 \right)^{1/2} (\Pr(A \text{ succeeds}))^{1/2} =$$
$$(|\mathcal{R}| \Pr(A \text{ succeeds}))^{1/2} \frac{(1 + |\mathcal{R}|\delta_2(G)^2)^{m/2} - 1}{(1 + |\mathcal{R}|\delta_2(G)^2)^{1/2} - 1} \delta_2(G) \ .$$

□