Projective Coordinates Leak

David Naccache[1], Nigel P. Smart[2], and Jacques Stern[3]

[1] Gemplus Card International,
Applied Research & Security Centre,
34 rue Guynemer,
Issy-les-Moulineaux, F-92447,
France
david.naccache@gemplus.com

[2] Department of Computer Science,
University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol, BS8 1UB,
United Kingdom
nigel@cs.bris.ac.uk

[3] École Normale Supérieure,
Département d'Informatique,
45 rue d'Ulm,
F-75230, Paris 05,
France.
jacques.stern@ens.fr

Abstract. Denoting by $P = [k]G$ the elliptic-curve double-and-add multiplication of a public base point G by a secret k, we show that allowing an adversary access to the projective representation of P, obtained using a particular double and add method, may result in information being revealed about k.

Such access might be granted to an adversary by a poor software implementation that does not erase the Z coordinate of P from the computer's memory or by a computationally-constrained secure token that sub-contracts the affine conversion of P to the external world.

¿From a wider perspective, our result proves that the choice of representation of elliptic curve points *can reveal* information about their underlying discrete logarithms, hence casting potential doubt on the appropriateness of blindly modelling elliptic-curves as generic groups.

As a conclusion, our result underlines the necessity to sanitize Z after the affine conversion or, alternatively, randomize P before releasing it out.

1 Introduction

There are various systems of projective coordinates that are used in conjunction with elliptic curves: the usual (classical) system replaces the affine coordinates

(x, y) by any triple $(X, Y, Z) = (\lambda x, \lambda y, \lambda)$, where $\lambda \neq 0$ is an element of the base field.

¿From such a (X, Y, Z), the affine coordinates are computed back as

$$\left(x = \frac{X}{Z}, y = \frac{Y}{Z}\right) = \text{Affine}(X, Y, Z)$$

A variant of the above, often called Jacobian Projective coordinates, replaces the affine coordinates (x, y) by any triple $(\lambda^2 x, \lambda^3 y, \lambda)$, where λ is a non zero element of the base field. From (X, Y, Z), the affine coordinates are computed as

$$\left(x = \frac{X}{Z^2}, y = \frac{Y}{Z^3}\right) = \text{Affine}(X, Y, Z)$$

These coordinates are widely used in practice, see for example [1] and [4].

This paper explores the following question:

Denoting by $P = [k]G$ the elliptic-curve multiplication of a public base point G by a secret k, does the projective representation of P result in information being revealed about k?

¿From a practical perspective access to P's Z coordinate might stem from a poor software implementation that does not erase the Z coordinate of P from the computer's memory or caused by a computationally-constrained secure token that sub-contracts the affine conversion of P to the external world.

We show that information may leaks-out and analyse the leakage in two different settings: Diffie-Hellman key exchange and Schnorr signatures.

Moreover, our paper seems to indicate that *point representation matters*: The generic group model is often used to model elliptic curve protocols, see [2], [10], [11]. In this model one assumes that the representation of the group elements gives no benefit to an adversary. This approach allows cryptographic schemes built from elliptic curves to be supported by some form of provable security. However, it has some pitfalls. In [11], it was shown that using encodings which do not adequately distinguish an elliptic curve from its opposite, as done in ECDSA, open the way to potential flaws in the security proofs. In this paper we show that using projective coordinates to represent elliptic curve points rather than affine coordinates may leak some information to an attacker. Thus, we can conclude that modelling elliptic curves as generic groups is not appropriate in this case, so that the generic model methodology only applies under the assumption that affine points are made available to an external viewer/adversary of the protocol.

We note that our results imply that projective coordinates should be used with care when they could be made available to an adversary. Our results do not however imply that using projective coordinates for *internal* calculations has any security implications.

2 Elliptic Curve Addition Formulae

In the following, we will restrict our attention to elliptic curves over fields of large prime characteristic. We will also focus on projective coordinates of the second kind (the situation being quite similar *mutatis mutandis*, in the other cases).

In our prime field case, the reduced equation of the curve $\mathcal{C}$ is:

$$y^2 = x^3 + ax + b \mod p$$

Jacobian projective coordinates yield the equation:

$$Y^2 = X^3 + aXZ^4 + bZ^6 \mod p$$

Projective coordinates allow a smooth representation of the infinity point $\mathcal{O}$ on the curve: $(0, 1, 0)$ in the first system, $(1, 1, 0)$ in the other. They also provide division-free formulae for addition and doubling.

Standard (affine) addition of two distinct elliptic curve points, (x_0, y_0) and (x_1, y_1) yields (x_2, y_2), with:

$$x_2 = \left(\frac{y_1 - y_0}{x_1 - x_0}\right)^2 - x_0 - x_1$$

Note that $x_1 - x_0$ equals:

$$\frac{X_1}{Z_1^2} - \frac{X_0}{Z_0^2} = \frac{W}{(Z_0 Z_1)^2}$$

where W is $X_1 Z_0^2 - X_0 Z_1^2$. ¿From this it readily follows, that $(W Z_0 Z_1)^2 x_2$ is a polynomial in $X_0, Y_0, Z_0, X_1, Y_1, Z_1$, since the further factors coming from Z_0 and Z_1 cancel the denominators for x_0 and x_1.

The affine coordinate y_2 is given by:

$$y_2 = -y_0 + \left(\frac{y_1 - y_0}{x_1 - x_0}\right)(x_0 - x_2)$$

Expanding in projective coordinates yields a denominator equal to $W^3 Z_0^3 Z_1^3$. Thus, $(W Z_0 Z_1)^3 y_2$ is a polynomial in X_0, Y_0, Z_0, X_1, Z_1. Finally, we see that setting:

$$Z_2 = W Z_0 Z_1$$

we can obtain division-free formulae. Such formulae are given in [4] and [1], and we simply reproduce them here:

$$\begin{array}{llll} U_0 \leftarrow X_0 Z_1^2, & S_0 \leftarrow Y_0 Z_1^3, & U_1 \leftarrow X_1 Z_0^2, & S_1 \leftarrow Y_1 Z_0^3, \\ W \leftarrow U_0 - U_1, & R \leftarrow S_0 - S_1, & T \leftarrow U_0 + U_1, & M \leftarrow S_0 + S_1, \\ Z_2 \leftarrow W Z_0 Z_1, & X_2 \leftarrow R^2 - TW^2, & V \leftarrow TW^2 - 2X_2, & 2Y_2 \leftarrow VR - MW^3. \end{array}$$

There is a similar analysis for doubling; again, we simply provide the corresponding formulae:

$$\begin{array}{lll} M \leftarrow 3X_1^2 + aZ_1^4, & Z_2 \leftarrow 2Y_1 Z_1, & S \leftarrow 4X_1 Y_1^2, \\ X_2 \leftarrow M^2 - 2S, & T \leftarrow 8Y_1^4, & Y_2 \leftarrow M(S - X_2) - T. \end{array}$$

3 The Attack

Throughout this section we let G be an element of prime order r on an elliptic curve $\mathcal{C}$ over a prime field, given by its regular coordinates (x_G, y_G). Let k be a secret scalar and define $P = [k]G$. Let (X, Y, Z) be Jacobian projective coordinates for P, computed by the formulae introduced in Section 2, when the standard double-and-add algorithm is used.

3.1 Grabbing a few bits of k

Let t be a small integer and guess the last t bits of k. Once this is done, it is possible to compute a set of candidates for the coordinates of the sequence of intermediate values handled by the double-and-add algorithm while processing k's t trailing bits (appearing at the end of the algorithm). This is achieved by 'reversing' computations: reversing doubling is halving, *i.e.* by reversing the formulae for doubling ; reversing an addition amounts to subtracting G. Thus, we obtain a set of sequences,

$$\{s_1, s_2, \ldots, s_m\} \quad \text{where} \quad s_j = \{M_0^{(j)} \dashrightarrow M_1^{(j)} \dashrightarrow \cdots \dashrightarrow M_\ell^{(j)}\}$$

of intermediate points, with $M_\ell^{(j)} = P$. Let $M_i = (x_i, y_i)$ in affine coordinates. The corresponding projective coordinates which occur we denote by (X_i, Y_i, Z_i). There are two cases:

- When the step $M_i \dashrightarrow M_{i+1}$ is an addition, we have

  $$Z_{i+1} = (X_i - x_G Z_i^2) Z_i \quad \text{which yields} \quad \frac{Z_{i+1}}{Z_i^3} = (x_i - x_G)$$

  Here, we need to compute a cubic root to get Z_i from Z_{i+1}. This is impossible in some cases when $p \equiv 1 \mod 3$, and when possible, it leads to one of three possible Z_i values. When $p \equiv 2 \mod 3$ taking the cubic root is always possible and leads to a unique value of Z_i. In either case once a set of possible values of Z_i are determined from Z_{i+1} we can obtain X_i and Y_i.
- When the step $M_i \dashrightarrow M_{i+1}$ is a doubling, we have

  $$Z_{i+1} = 2Y_i Z_i \quad \text{which yields} \quad \frac{Z_{i+1}}{Z_i^4} = 2y_i$$

  Here, we need to compute a fourth root to get Z_i from Z_{i+1}, which is impossible in some cases. Assume for example that $p \equiv 3 \mod 4$. Then extracting a fourth root is possible for one half of the inputs and, when possible, yields two values. When $p \equiv 1 \mod 4$ then this is possible in around one quarter of all cases and yields four values.

We can now take advantage of the above observation to learn a few bits of k.

More precisely, we observe that, with probability at least $1/2$, one can spot values of k for which the least significant trailing bit is one. Suppose we consider such a k and make the wrong guess that the last bit is zero. This means that the final operation $M_{\ell-1} \dashrightarrow M_\ell$ is a doubling. The error can be spotted when the value

$$\frac{Z_\ell}{2y_{\ell-1}}$$

is not a fourth power, which happens with probability at most 1/2. We can then iterate this to (potentially) obtain a few further bits of k. In the case of the least significant bit being zero a similar analysis can be performed.

3.2 Applicability to Different Coordinate Systems

Consider Jacobian projective coordinates:

$$(X,Y) \mapsto (\lambda^2 X, \lambda^3 Y, \lambda Z),$$

over a field $\mathbb{F}_q$ of characteristic $q > 3$. For a point $P = (x,y) \in \mathcal{C}(\mathbb{F}_q)$ let S_P denote the set of all equivalent projective representations

$$S_P = \{(\lambda^2 x, \lambda^3 y, \lambda) : \lambda \in \mathbb{F}_q^*\}.$$

The standard addition formulae for computing $P + Q$, for a fixed value of Q (by fixed we mean a fixed projective representation of Q, including an affine representation of Q) gives a map

$$\Psi_{P,P+Q} : S_P \longrightarrow S_{P+Q}.$$

The doubling formulae for Jacobian projective coordinates also gives us a map

$$\Psi_{P,[2]P} : S_P \longrightarrow S_{[2]P}.$$

The crucial observations from the previous subsection are summarized in the following Lemma

Lemma 1. *The following holds, for Jacobian projective coordinates in large prime characteristics:*

$$\begin{array}{llll}
\text{If } q \equiv 1 \mod 3 & \text{then } \Psi_{P,P+Q} & \text{is a } 3 \rightsquigarrow 1 \text{ map.} \\
\text{If } q \equiv 2 \mod 3 & \text{then } \Psi_{P,P+Q} & \text{is a } 1 \rightsquigarrow 1 \text{ map.} \\
\text{If } q \equiv 1 \mod 4 & \text{then } \Psi_{P,[2]P} & \text{is a } 4 \rightsquigarrow 1 \text{ map.} \\
\text{If } q \equiv 3 \mod 4 & \text{then } \Psi_{P,[2]P} & \text{is a } 2 \rightsquigarrow 1 \text{ map.}
\end{array}$$

Note: It is easy given an element in the image of either $\Psi_{P,P+Q}$ or $\Psi_{P,[2]P}$ to determine whether it has pre-images, and if so to compute all of them.

The attack is then simply to consider when a point could have arisen from an application of $\Psi_{P,P+Q}$ or $\Psi_{P,[2]P}$ and if so to compute all the pre-images and then recurse. The precise tests one applies at different points will depend on the precise exponentiation algorithm implemented by the attacked device, a subject we shall return to in a moment.

For the sake of completeness we present in the following *lemmata* similar results for other characteristics and other forms of projective representation. We concentrate on the most common and the most used coordinate systems and keep the same conventions and notation as above:

Lemma 2. *The following holds, for classical projective coordinates on elliptic curves over fields of large prime characteristic:*

$$\begin{array}{lll} \textit{If } q \equiv 1 \mod 4 & \textit{then } \Psi_{P,P+Q} & \textit{is a } 4 \leadsto 1 \textit{ map.} \\ \textit{If } q \equiv 3 \mod 4 & \textit{then } \Psi_{P,P+Q} & \textit{is a } 2 \leadsto 1 \textit{ map.} \\ \textit{If } q \equiv 1 \mod 3 & \textit{then } \Psi_{P,[2]P} & \textit{is a } 6 \leadsto 1 \textit{ map.} \\ \textit{If } q \equiv 2 \mod 3 & \textit{then } \Psi_{P,[2]P} & \textit{is a } 2 \leadsto 1 \textit{ map.} \end{array}$$

Lemma 3. *The following holds, for Jacobian projective coordinates on elliptic curves over fields of characteristic two:*

$$\begin{array}{lll} \textit{If } q \equiv 1 \mod 3 & \textit{then } \Psi_{P,P+Q} & \textit{is a } 3 \leadsto 1 \textit{ map.} \\ \textit{If } q \equiv 2 \mod 3 & \textit{then } \Psi_{P,P+Q} & \textit{is a } 1 \leadsto 1 \textit{ map.} \\ & \forall q \quad \Psi_{P,[2]P} & \textit{is a } 1 \leadsto 1 \textit{ map.} \end{array}$$

Lemma 4. *The following holds, for López-Dahab projective coordinates [6] on elliptic curves over fields of characteristic two:*

$$\begin{array}{lll} \textit{If } q \equiv 1 \mod 3 & \textit{then } \Psi_{P,[2]P} & \textit{is a } 3 \leadsto 1 \textit{ map.} \\ \textit{If } q \equiv 2 \mod 3 & \textit{then } \Psi_{P,[2]P} & \textit{is a } 1 \leadsto 1 \textit{ map.} \\ & \forall q \quad \Psi_{P,P+Q} & \textit{is a } 1 \leadsto 1 \textit{ map.} \end{array}$$

4 Application: Breaking Projective Schnorr Signatures

Assume now that one wishes to use the protocol described in Figure 1, mimicking Schnorr's basic construction [12]. The algorithm is a natural division-free version of Schnorr's original scheme, and might hence appear both safe and computationally attractive.

It should be stressed that while we are *not* aware of any suggestion to use this variant in practice it is still not evident, at a first glance, why this algorithm could be insecure.

We show how to attack this scheme using the observations from the previous subsection. This is based on recent work by Howgrave-Graham, Smart [3], Nguyen and Shparlinski [7].

¿From a sample of N signatures, the attacker obtains around $\frac{N}{2^{2t}}$ signatures for which he knows that the t low order bits of the hidden nonce k are ones. Next, for each such k, he considers the relation:

$$d + xH(m, P_X, P_Y, P_Z) = k \mod r$$

Using the information he has, the attacker rewrites the above as:

$$d - (2^t - 1) + xH(m, P_X, P_Y, P_Z) = k - (2^t - 1) \mod r$$

PARAMETERS AND KEYS	
	An elliptic-curve $\mathcal{C}$
	$G \in_R \mathcal{C}$ of order r
	A collision-resistant hash-function $H : \{0,1\}^* \longrightarrow \mathbb{Z}_r^\star$
Private	$x \in_R \mathbb{Z}_r^\star$
Public	$Q \leftarrow [x]G$
SIGNATURE GENERATION	
	Pick $k \in_R \mathbb{Z}_r^\star$
	Compute $(P_X, P_Y, P_Z) \leftarrow [k]G = \text{DoubleAdd}(k, G)$
	$d \leftarrow k - x \times H(m, P_X, P_Y, P_Z) \mod r$
	If $d = 0$ or $H(m, P_X, P_Y, P_Z) = 0$ resume signature generation
	Output $\{P_X, P_Y, P_Z, d\}$ as the signature of m
SIGNATURE VERIFICATION	
	$P \leftarrow [d]G + [H(m, P_X, P_Y, P_Z)]Q$
	If $P \neq \text{Affine}((P_X, P_Y, P_Z))$ or $d \notin \mathbb{Z}_r^\star$ output invalid
	else output valid

Fig. 1. Division-Free Projective Schnorr Signatures

Dividing by 2^t, he gets a final relation:

$$a + bx = u \mod r$$

where a, b are known but x is unknown as well as u. Still the attacker knows that u is small ($\leq \frac{r}{2^t}$). When the attacker has $n \approx \frac{N}{2^{2t}}$ such relations, he writes

$$\mathbf{a} + \mathbf{b}x = \mathbf{u} \mod r$$

and considers the lattice $L = (\mathbf{b})^\perp$, consisting of all integer vectors orthogonal to $\mathbf{b}$ and applies lattice reduction. Let $\mathbf{\Lambda}$ be an element of L with small Euclidean norm. We have:

$$\mathbf{\Lambda}(\mathbf{a}) = \mathbf{\Lambda}(\mathbf{u}) \mod r$$

Now, the norm of the right-hand side is bounded by $||\mathbf{\Lambda}||||\mathbf{u}||$, which is $\leq ||\mathbf{\Lambda}||\frac{r}{2^t}\sqrt{n}$. The order of $||\mathbf{\Lambda}||$ is $r^{\frac{1}{n}}$ and, for n large enough and t not too small, this estimate provides a bound for the right-hand side $< r/2$. Thus, the modular equations are actual equations over the integers:

$$\mathbf{\Lambda}(\mathbf{a}) \mod r = \mathbf{\Lambda}(\mathbf{u})$$

The attacker can hope for at most $n-1$ such relations, since L has dimension $n-1$. This defines $\mathbf{u}$ up to the addition of an element from a one-dimensional lattice. The correct value is presumably the element in this set closest to the origin. Once $\mathbf{u}$ has been found, the value of x follows.

PARAMETERS	
Input	$k \in \mathbb{Z}_r^\star, G \in \mathcal{C}$
Output	$P \leftarrow [k]G$
ALGORITHM DoubleAdd(k, G)	
	$P \leftarrow \mathcal{O}$
	for $j = \ell - 1$ downto 0:
	$\quad P \leftarrow [2]P$
	$\quad$ if $k_j = 1$ then $P \leftarrow P + G$
	return(P)

Fig. 2. Double-and-Add Exponentiation

Lattice reduction experiments reported in [7] show that, with elliptic curves of standard dimensions, the attack will succeed as soon as t reaches 5 digits. The deep analysis of Nguyen and Shparlinski, shows that the significant theoretical bound is related to $\sqrt{\log r}$.

5 Practical Experiments

The double-and-add exponentiation's case is the simplest to analyse: given the projective representation of the result P, we can try and 'unwind' the algorithm with respect to the fixed point G.

In other words, we can check whether there is a value P' such that

$$\Psi_{P',P'+G}(P') = P$$

and if so compute all the pre-images P'. Then for all pre-images P' we can check whether this was the result of a point doubling. We also need to check whether P itself was the output of a point doubling. This results in a backtracking style algorithm which investigates all possible execution paths through the algorithm.

There are two factors at work here. For each testing of whether $\Psi_{P,P+G}$ (resp. $\Psi_{P,[2]P}$) was applied we have a representation-dependent probability of $\mathfrak{p}$ (from the above *lemmata*), this acts in the attacker's favour. However, each success for this test yields $1/\mathfrak{p}$ pre-images, which increases the attacker's workload. The result is that, while practical, the attack against the double-and-add algorithm is not as efficient as one might initially hope.

We ran one thousand experiments in each prime characteristic modulo 12. Table 1 presents the success of determining the parity of the secret exponent. One should interpret the entries in the table as follows: For example with $q \equiv 5 \pmod{12}$, we found that in 71 percent of all cases in which k was even we where able to determine this using the above backtracking algorithm. This means that

PARAMETERS	
Input	$k \in \mathbb{Z}_r^\star, G \in \mathcal{C}$
Output	$P \leftarrow [k]G$
PRECOMPUTATION	
	$G_1 \leftarrow G$
	$G_2 \leftarrow [2]G$
	for $j = 1$ to $2^{r-2} - 1$:
	$\quad G_{2j+1} \leftarrow G_{2j-1} + G_2$
	$P \leftarrow G_{k_\ell - 1}$
EXPONENT ENCODING	
	set $k = \sum_{i=0}^{\ell-1} k_i 2^{e_i}$
	with $e_{i+1} - e_i \geq r$ and $k_i \in \{\pm 1, \pm 3, \ldots, \pm 2^{r-1} - 1\}$
ALGORITHM SlidingWindow(k, G)	
	for $j = \ell - 2$ downto 0:
	$\quad P \leftarrow [2^{e_{j+1} - e_j}]P$
	$\quad$ if $\ell_j > 0$ then $P \leftarrow P + G_{k_j}$ else $P \leftarrow P + G_{-k_j}$
	$P \leftarrow [2^{e_0}]P$
	return(P)

Fig. 3. Signed Sliding Window Exponentiation

in these cases the execution path which started with assuming P was the output of a point addition was eventually determined to be invalid.

Table 1. Probability of Determining the Secret's Parity Using Double-and-Add Exponentiation

$q \mod 12$	1	5	7	11
Pr[parity determined$\|k$ even]	0.98	0.71	0.80	0.50
Pr[parity determined$\|k$ odd]	0.95	0.74	0.50	0.47
Pr[parity determined]	0.96	0.72	0.65	0.48

Only in the cases $q \equiv 1 \mod 12$ and $q \equiv 7 \mod 12$ did we have any success in determining the value of the secret exponent modulo 8 precisely (around 50 percent of the time when $q \equiv 1 \mod 12$ and 8 percent of the time when $q \equiv 7 \mod 12$).

We did a similar experiment using the signed sliding window method, with a window width of 5 (see also Algorithm IV.7 of [1]) assuming that the precomputed table of multiples of the base point is known to the attacker. In this case we had a much lower probability of determining the parity, but could still determine the value of the exponent modulo 32 in a significant number of cases (Table 2).

Table 2. Probability of Determining the Secret's Parity Using Signed Sliding Window Exponentiation

$q \bmod 12$	1	5	7	11
Pr[parity determined$\vert k$ even]	0.86	0.00	0.05	0.00
Pr[parity determined$\vert k$ odd]	0.81	0.75	0.49	0.53
Pr[parity determined]	0.81	0.37	0.27	0.26
Pr[$k \bmod 32$ determined]	0.42	0.01	0.01	0.00

Note that this means that if $q \equiv 1 \bmod 12$ then we will be successful in determining the full private key for the division free signature algorithm of Section 4 using lattice reduction.

6 Thwarting The Attack

There is a simple trick that avoids the attacks described in the previous sections. It consists in randomly replacing the output (X, Y, Z) of the computation by $(X, \epsilon Y, \epsilon Z)$, with $\epsilon = \pm 1$. This makes it impossible for an attacker to spot projective coordinates, which cannot be obtained by squaring. It should be underlined that this countermeasure (that we regard as a challenge for the research community) thwarts *our* specific attack but does not lend itself to a formal security proof. Note, such a defence only appears to need to be done at the end of the computation as our attack model assume the attacker does not obtain any intermediate points from the multiplication algorithm.

A more drastic method replaces (X, Y, Z) by $(\lambda^2 x, \lambda^3 y, \lambda)$, where λ is randomly chosen among the non zero elements of the base field (with ordinary projective coordinates, one uses $(\lambda x, \lambda y, \lambda)$). This method provides a randomly chosen set of projective coordinates for the result and, therefore, cannot leak additional information.

With this new protection, the division-free signature scheme of Section 4 can be shown to be secure in the random oracle model, against adaptive attackers trying to achieve existential forgery. We outline the proof. As usual (see [9]), one uses the attacker to solve the discrete logarithm problem (here, on $\mathcal{C}$). The public key of the scheme is set to Q, the curve element for which we want to compute the discrete logarithm in base G. Signature queries are answered by randomly creating $P = [d]G + [h]Q$, picking random projective coordinates for P, say (X, Y, Z) and setting the hash value of $\{m, X, Y, Z\}$ as any element $= h \bmod r$. Thus fed, the attacker should create a forged message signature pair, with significant probability. We let m be the corresponding message and $\{X, Y, Z, d\}$

be the signature. With significant probability, $\{m, X, Y, Z\}$ is queried from the random oracle. Replaying the attack with a different answer modulo r to this question, one gets, with significant probability, another forgery $\{m, X, Y, Z, d'\}$, with h replaced by h'. ¿From the relation

$$[d]G + [h]Q = [d']G + [h']Q$$

one finally derives the discrete logarithm of Q.

References

1. I.F. Blake, G. Seroussi and N.P. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.
2. D. Brown, *Generic Groups, Collision Resistance, and ECDSA*, ePrint Report 2002/026, `http://eprint.iacr.org/`.
3. N.A. Howgrave-Graham and N.P. Smart, *Lattice attacks on digital signature schemes*, *Designs, Codes and Cryptography*, **23**, pp. 283–290, 2001.
4. IEEE 1363, IEEE standard specifications for public key cryptography, 2000.
5. A. Joux and J. Stern, *Lattice Reduction: a Toolbox for the Cryptanalyst*, In *Journal of Cryptology*, vol. 11, pp. 161–186, 1998.
6. J. López and R. Dahab, *Improved algorithms for elliptic curve arithmetic in $GF(2^n)$*, In *Selected Areas in Cryptography - SAC'98*, Springer-Verlag LNCS 1556, pp. 201–212, 1999.
7. P. Nguyen and I. Shparlinski, *The Insecurity of the Digital Signature Algorithm with Partially Known Nonces*, In *Journal of Cryptology*, vol. 15, pp. 151–176, 2002.
8. P. Nguyen and J. Stern, *The hardness of the subset sum problem and its cryptographic implications*, In *Advances in Cryptology CRYPTO'99*, Santa Barbara, Lectures Notes in Computer Science 1666, pp. 31–46, Springer-Verlag, 1999.
9. D. Pointcheval and J. Stern, *Security Arguments for Digital Signatures and Blind Signatures*, In *Journal of Cryptology*, vol. 13, pp. 361–396, 2000.
10. N. P. Smart, *The Exact Security of ECIES in the Generic Group Model* In B. Honary (Ed.), Cryptography and Coding 8-th IMA International Conference Cirencester, LNCS 2260, Springer Verlag, pp. 73–84, 2001.
11. J. Stern, D. Pointcheval, J. Malone-Lee and N. P. Smart, *Flaws in Applying Proof Methodologies to Signature Schemes*, In *Advances in Cryptology CRYPTO'02*, Santa Barbara, Lectures Notes in Computer Science 2442, pp. 93–110, Springer-Verlag, 2002.
12. C. P. Schnorr, *Efficient Signature Generation by Smart Cards*, In *Journal of Cryptology*, vol. 4, pp. 161–174, 1991.
13. U.S. Department of Commerce, National Institute of Standards and Technology. Digital Signature Standard. Federal Information Processing Standard Publication 186, 1994.