# Lattice Attacks on DSA Schemes Based on Lagrange's Algorithm

Konstantinos Draziotis[1] and Dimitrios Poulakis[2]

[1] Department of Informatics, Aristotle University of Thessaloniki, P.O. Box 114, Thessaloniki 54124, Greece
drazioti@csd.auth.gr
[2] Department of Mathematics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece
poulakis@math.auth.gr

**Abstract.** Using Lagrange's algorithm for the computation of a basis of a 2-dimensional lattice formed by two successive minima, we present some attacks on DSA and ECDSA which permit us, under some assumptions, to compute the secret key of the scheme provided that one or two signed messages are given.

MSC 2010: 94A60, 11T71, 11Y16.

**Keywords:** Public Key Cryptography, Digital Signature Algorithm, Elliptic Curve Digital Signature Algorithm, Lagrange's Algorithm, Lattice Reduction, Lattice attack.

## 1 Introduction

*The signature schemes DSA and ECDSA.* In 1985, T. ElGamal published the first digital signature scheme based on the discrete logarithm problem in the finite prime fields $\mathbb{F}_p$ [5]. Since then several variants of this scheme have been proposed. In 1991, the U.S. government's National Institute of Standards and Technology (NIST) proposed an efficient variant of the ElGamal signature scheme known as DSA, for Digital Signature Algorithm [6,12,14]. In 1998, an elliptic curve analogue called Elliptic Curve Digital Signature Algorithm (ECDSA) was proposed and standardized [8,11,12].

First, let us summarize DSA. The signer chooses a prime $p$ of size between 1024 and 3072 bits with increments of 1024, as recommended in FIPS 186-3 [6, page 15]. Also he chooses a prime $q$ of size 160, 224 or 256 bits, with $q|p-1$ and a generator $g$ of the unique order $q$ subgroup $G$ of $\mathbb{F}_p^*$. Furthermore, he selects a random integer $a \in \{1, \ldots, q-1\}$ and computes $R = g^a \bmod p$. The public key of the signer is $(p, q, g, R)$ and his private key $a$. He also publishes a hash function $h : \{0,1\}^* \to \{0, \ldots, q-1\}$. To sign a message $m \in \{0,1\}^*$, he selects a random number $k \in \{1, \ldots, q-1\}$ which is the ephemeral key, and computes

$$r = (g^k \bmod p) \bmod q, \qquad s = k^{-1}(h(m) + ar) \bmod q.$$

T. Muntean, D. Poulakis, and R. Rolland (Eds.): CAI 2013, LNCS 8080, pp. 119–131, 2013.

The signature of $m$ is the pair $(r, s)$. The verification of the signature is performed by checking

$$r = ((g^{s^{-1}h(m) \bmod q} R^{s^{-1}r \bmod q}) \bmod p) \bmod q.$$

For the ECDSA the signer selects an elliptic curve $E$ over the finite prime field $\mathbb{F}_p$, a point $P \in E(\mathbb{F}_p)$ with order a prime $q$ of size at least 160 bits. According to FIPS 186-3, the prime $p$ must be in the set $\{160, 224, 256, 512\}$. Further, he chooses a random integer $a \in \{1, \ldots, q-1\}$ and computes $Q = aP$. The public key of the signer is $(E, p, q, P, Q)$ and his private key $a$. He also publishes a hash function $h : \{0,1\}^* \to \{0, \ldots, q-1\}$. To sign a message $m$, he selects a random number $k \in \{1, \ldots, q-1\}$ which is the ephemeral key and computes $kP = (x, y)$ (where $x$ and $y$ are regarded as integers between 0 and $p-1$). Next, he computes

$$r = x \bmod q \quad \text{and} \quad s = k^{-1}(h(m) + ar) \bmod q.$$

The signature of $m$ is $(r, s)$. For its verification one computes

$$u_1 = s^{-1}h(m) \bmod q, \quad u_2 = s^{-1}r \bmod q, \quad u_1 P + u_2 Q = (x_0, y_0).$$

He accepts the signature if and only if $r = x_0 \bmod q$.
The security of the two systems is relied on the assumption that the only way to forge the signature is to recover either the secret key $a$, or the ephemeral key $k$ (in this case is very easy to compute $a$). Thus, the parameters of these systems were chosen in such a way that the computation of discrete logarithms is computationally infeasible.

*Related Work.* A very important tool for attacking public-key cryptosystems are the lattices and the so-called LLL reduction method [13]. Attacks to DSA and to ECDSA based on these techniques and using he equality $s = k^{-1}(h(m) + ar) \bmod q$ are described in [1,3,10,16,17,18,19]. In [1], it was shown that the DSA secret key $a$ can be computed provided the ephemeral key $k$ is produced by Knuth's linear congruential generator with known parameters, or variants. In [10] the authors pointed out that Babai's method can be used heuristically in order to recover the secret key $a$, if enough signatures and some bits of the ephemeral keys are known. A polynomial time attack which computes the DSA secret key $a$ is proposed in [15], in case the following holds: The size of $q$ is not too small compared with $p$, the probability of collisions for the hash function is not too large compared to $1/q$ and for a polynomially bounded number of messages, about $\lfloor \sqrt{\log_2 q} \rfloor$ of the least significant bits of the ephemeral keys are known. This attack is adapted to the case of ECDSA [18]. In [3], the authors compute, using the LLL reduction method, two short vectors of a three-dimensional lattice and in case where the second shortest vector is sufficiently short, they deduce two lines which intersect in $(a, k)$, provided that $a$ and $k$ are sufficiently small. Finally, in [19], an attack is described in case where the secret and the ephemeral key of a signed message or their modular inverses are sufficiently small.

*Our Contribution.* In this paper we present some attacks on DSA and ECDSA taking advantage of the equation $s = k^{-1}(h(m) + ar) \bmod q$. We use a two-dimensional lattice $\mathcal{L}$ defined by a signed message. Lagrange Lattice Reduction

algorithm, provide us with a basis of $\mathcal{L}$ formed by two successive minima $\mathbf{b}_1, \mathbf{b}_2$. Using this basis we construct two linear polynomials $f_i(x, y)$ such that $(a, k)$ is the intersection point of two straight lines of the form $f_i(x, y) = c_i q$, where $c_i \in \mathbb{Z}$ $(i = 1, 2)$. If $a$ and $k$ are sufficiently small, then $c_i$ belong to a small set and so we can compute the secret key $a$ in polynomial time. Similar attacks hold for the pairs $(k^{-1} \bmod q, k^{-1}a \bmod q)$ and $(a^{-1} \bmod q, a^{-1}k \bmod q)$.

If two signed messages with ephemeral keys $k_1$ and $k_2$ are available, then we obtain an equation of the same form relating $k_1$ and $k_2$ and so, earlier statements about secret and ephemeral keys also apply to $k_1$ and $k_2$. In case we know some bits of the secret and ephemeral keys and their sum is almost the half of sum of the bits of secret and ephemeral keys, then we can compute efficiently the secret key. Similar results also hold for all the aforementioned cases.

The paper is organized as follows. In Section 2, we recall some basic results about Lagrange lattice reduction algorithm. Our attacks are presented in Sections 3 - 10. An example is given in Section 11 and Section 12 concludes the paper.

## 2 Lagrange Lattice Reduction

Let $\mathbb{R}^n$ denote the $n$-dimensional real Euclidean space. The *inner product* of two elements $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$ of $\mathbb{R}^n$ is defined to be the quantity $\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + \cdots + u_n v_n$ and the *Euclidean norm* of a vector $\mathbf{v} = (v_1, \ldots, v_n)$ the quantity $\|\mathbf{v}\| = (v_1^2 + \cdots + v_n^2)^{1/2}$.

Let $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be a basis of $\mathbb{R}^n$. A *n-dimensional lattice* spanned by $B$ is the set

$$\mathcal{L} = \{z_1 \mathbf{b}_1 + \cdots + z_n \mathbf{b}_n / \ z_1, \ldots, z_n \in \mathbb{Z}\}.$$

If $\mathbf{b}_i = (b_{i1}, \ldots, b_{in})$ $(i = 1, \ldots, n)$, then the *determinant* $\det\mathcal{L}$ of $\mathcal{L}$ is the absolute value of the determinant whose $(i, j)$ element is $b_{i,j}$.

Let $\mathcal{L} \subset \mathbb{R}^2$ be a 2-dimensional lattice. We say that a basis $\{\mathbf{v}_1, \mathbf{v}_2\}$ of $\mathcal{L}$ is formed by two *successive minima* of $\mathcal{L}$, if $\mathbf{v}_1$ is the shortest nonzero vector of $\mathcal{L}$ and $\mathbf{v}_2$ is the shortest vector of $\mathcal{L}$ among all vectors of $\mathcal{L}$ linearly independent from $\mathbf{v}_1$. In this section we recall an algorithm due to Lagrange (1773) which computes a basis of $\mathcal{L}$ formed by two successive minima of $\mathcal{L}$. This algorithm was also described later by Gauss (1801), and is often erroneously called Gauss' algorithm. If $x \in \mathbb{R}$, then we set $\lfloor x \rceil = \lfloor x + 0.5 \rfloor$.

LAGRANGE'S ALGORITHM
*Input:* A basis $\{\mathbf{v}_1, \mathbf{v}_2\}$ of $\mathcal{L}$.
*Output:* A basis $\{\mathbf{b}_1, \mathbf{b}_2\}$ formed by two successive minima of $\mathcal{L}$.
Loop
  If $\|\mathbf{v}_2\| < \|\mathbf{v}_1\|$, then swap $\mathbf{v}_1$ and $\mathbf{v}_2$.
  Compute $m = \lfloor \mathbf{v}_1 \cdot \mathbf{v}_2 / \|\mathbf{v}_1\|^2 \rceil$.
  If $m = 0$, return the basis vectors $\mathbf{v}_1$ and $\mathbf{v}_2$.
  Replace $\mathbf{v}_2$ with $\mathbf{v}_2 - m\mathbf{v}_1$.
Continue Loop

**Theorem 1.** *The above algorithm computes a basis* $\{\mathbf{b}_1, \mathbf{b}_2\}$ *formed by two successive minima of* $\mathcal{L}$ *in time* $O((\log M)^2)$ *bit operations, where* $M$ *is defined by* $M = \max\{\|\mathbf{v}_1\|, \|\mathbf{v}_2\|\}$. *If* $\theta$ *is the angle between* $\mathbf{b}_1$ *and* $\mathbf{b}_2$, *then* $\pi/3 \leq \theta \leq 2\pi/3$ *and* $|cos\theta| < \|\mathbf{b}_1\|/2\|\mathbf{b}_2\|$. *Further,* $\|\mathbf{b}_1\| \leq \sqrt{4/3}\ (\det \mathcal{L})^{1/2}$.

*Proof.* See [15], [9, Section 6.12.1] and [9, Theorem 6.25].

## 3 The DSA-ATTACK-1 Algorithm

Let $m$ be a message and $(r, s)$ its signature with DSA (resp. ECDSA). Then there is $k \in \{1, \ldots, q-1\}$ such that $r = (g^k \bmod p) \bmod q$ (resp. $kP = (x, y)$) and $s = k^{-1}(h(m) + ar) \bmod q$. We conserve these notations for the rest of the paper. In this section we present our first attack using the signed message.

Let $\mathbb{Z}_{(q)}$ be the subring of $\mathbb{Q}$ formed by the fractions $u/v$, where $u$ and $v$ are integers with $\gcd(u, v) = 1$ and $q \not| v$. If $x \in \mathbb{Z}$, then we denote by $\bar{x}$ the class of $x$ in $\mathbb{F}_q$. We have a ring homomorphism $\phi : \mathbb{Z}_{(q)} \to \mathbb{F}_q$ defined by $\phi(u/v) = \bar{u}(\bar{v})^{-1}$. We denote by $[u/v]_q$ the representative of the class $\phi(u/v)$ having the smallest absolute value. We have chosen this presentation for the elements of $\mathbb{F}_q$ since our method involves the absolute values of the unknown quantities which give the keys and our attacks are practical when these quantities are below a certain bound.

The following algorithm computes efficiently $a$ in case where $a$ and $k$ are sufficiently small.

DSA-ATTACK-1
*Input:* A signed message $(m, r, s)$ and integers $0 < X, Y < q$ with $X > \sqrt{q}$.

1. Compute $A = -rs^{-1} \bmod q$ and $B = -s^{-1}h(m) \bmod q$.
2. Compute, using Lagrange's algorithm, a basis formed by two successive minima $\mathbf{b}_1 = (b_{11}X, b_{12}Y)$ and $\mathbf{b}_2 = (b_{21}X, b_{22}Y)$ of the lattice $\mathcal{L}$ spanned by the vectors $(AX, Y)$ and $(qX, 0)$.
3. Compute $b_{i3} = b_{i2}B \bmod q$.
4. Compute the quantities

$$\beta_i = \left\lfloor \frac{b_{i3} + \sqrt{2}\ \|\mathbf{b}_i\|}{q} \right\rfloor \quad (i = 1, 2).$$

5. For every $c_1 \in \{0, \pm 1, \ldots, \pm\beta_1\}$ and $c_2 \in \{0, \pm 1, \ldots, \pm\beta_2\}$,
   (a) compute
$$x_0 = c_1 b_{22} - c_2 b_{12} + \frac{b_{23}b_{12} - b_{13}b_{22}}{q},$$
   (b) compute $g^{x_0} \bmod p$ (resp. $x_0 P$).
   (c) If $g^{x_0} = R \bmod p$ (resp. $x_0 P = Q$), then output $a = x_0 \bmod q$ and stop. Otherwise, output "Failure".

*Proof of Correctness.* We shall prove that this algorithm computes correctly $a$ in case $|[a]_q| < X$ and $|[k]_q| < Y$. For this purpose, we shall construct, using Lagrange's algorithm, a family of pairs of independent bivariate linear equations such that $([a]_q, [k]_q)$ is the common solution of one such pair, and hence we shall be able to determine it.

The determinant of the lattice $\mathcal{L}$ spanned by $(AX, Y)$ and $(qX, 0)$ is $\det \mathcal{L} = qXY$. By Theorem 1, Lagrange's algorithm gives a basis $\{\mathbf{b}_1, \mathbf{b}_2\}$ of $\mathcal{L}$ formed by two successive minima of $\mathcal{L}$ satisfying

$$2\,|cos\theta|\,\|\mathbf{b}_2\| < \|\mathbf{b}_1\| \le \sqrt{4/3}\,(qXY)^{1/2}, \tag{1}$$

where $\theta$ is the angle between $\mathbf{b}_1$ and $\mathbf{b}_2$.

Put $\mathbf{b}_i = (b_{i1}X, b_{i2}Y)$ $(i = 1, 2)$. Then there are $\lambda_{i1}$, $\lambda_{i2} \in \mathbb{Z}$ such that $b_{i1} = \lambda_{i1}q + \lambda_{i2}A$ and $b_{i2} = \lambda_{i2}$ $(i = 1, 2)$. Suppose that $q|\lambda_{i2}$. Then $\lambda_{i2} = \lambda'_{i2}q$, where $\lambda'_{i2} \in \mathbb{Z}$ $(i = 1, 2)$. We have

$$\|\mathbf{b}_i\|^2 = q^2\Big((\lambda_{i1} + \lambda'_{i2}A)^2X^2 + \lambda'^2_{i2}Y^2\Big) > \|(qX, 0)\|.$$

Since $X > \sqrt{q}$ and $0 < X, Y < q$, we have $Y/X < \sqrt{q}$ and so

$$\|(AX, Y)\|^2 \le ((q-1)^2 + q)X^2 < \|(qX, 0)\|^2.$$

Hence, we get

$$\|(AX, Y)\| < \|(qX, 0)\| < \|\mathbf{b}_i\| \quad (i = 1, 2)$$

which is a contradiction, since $\mathbf{b}_1$, $\mathbf{b}_2$ are two successive minima of $\mathcal{L}$. Hence, we have $\lambda_{i2} \not\equiv 0 \pmod q$. Further, by the Euclidean division, there are $\pi_i, b_{i3} \in \mathbb{Z}$ such that $\lambda_{i2}B = \pi_i q + b_{i3}$ and $0 < b_{i3} < q$ $(i = 1, 2)$.

Put $h_i(x, y) = b_{i1}x + b_{i2}y + b_{i3}$ $(i = 1, 2)$. We have

$$h_i(x, y) \equiv \lambda_{i2}(y + Ax + B) \pmod q \quad (i = 1, 2).$$

Since $\lambda_{i2} \not\equiv 0 \pmod q$, we deduce that $h_i(x, y)$ $(i = 1, 2)$ are not identical to zero over $\mathbb{F}_q$. Moreover, the congruence $[k]_q + A[a]_q + B \equiv 0 \pmod q$ implies $h_i([a]_q, [k]_q) \equiv 0 \pmod q$ $(i = 1, 2)$. On the other hand, we deduce

$$|h_i([a]_q, [k]_q)| \le b_{i3} + |\mathbf{b}_i \cdot ([a]_q/X, [k]_q/Y)| \le b_{i3} + \sqrt{2}\,\|\mathbf{b}_i\| \quad (i = 1, 2).$$

Setting

$$\beta_i = \left\lfloor \frac{b_{i3} + \sqrt{2}\,\|\mathbf{b}_i\|}{q} \right\rfloor \quad (i = 1, 2),$$

we have $h_i([a]_q, [k]_q) = c_i q$ where $|c_i| \in \{0, 1, \ldots, \beta_i\}$ $(i = 1, 2)$.

Since the vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are linearly independent, the lines defined by the equations $h_1(x, y) = c_1 q$ and $h_2(x, y) = c_2 q$ are not parallels. Thus there are $c_1$, $c_2$ as previously such that $([a]_q, [k]_q)$ is the intersection point of the lines $h_1(x, y) = c_1 q$ and $h_2(x, y) = c_2 q$. Hence we have

$$[a]_q = c_1 b_{22} - c_2 b_{12} + \frac{b_{23}b_{12} - b_{13}b_{22}}{q}, \qquad [k]_q = c_2 b_{11} - c_1 b_{21} + \frac{b_{13}b_{21} - b_{11}b_{23}}{q},$$

where $|c_i| \in \{0, 1, \ldots, \beta_i\}$ $(i = 1, 2)$.

*Time Complexity.* In Step 1, the application of the Extended Euclidean algorithm for the computation of $s^{-1} \bmod q$ and next the modular multiplications for the computations of $A$ and $B$ require $O((\log q)^2)$ bit operations. By Theorem 1, the complexity of Step 2 is $O((\log qXY)^2)$ bit operations. Step 3 and 4 need $O((\log q)^2)$ bit operations. Put $\beta_i^* = \max\{1, \beta_i\}$ $(i = 1, 2)$. Step 5 needs

$$O(\beta_1^*\beta_2^*(\log p)^2 \log(q\beta_1^*\beta_2^*))$$

bit operations, in case of DSA and

$$O(\beta_1^*\beta_2^* \log q \log(q\beta_1^*\beta_2^*))$$

bit operations and

$$O(\beta_1^*\beta_2^* \log q)$$

elliptic curve group operations, in case of ECDSA. Further, note that the bit complexity analysis can be improved by using the Shoenhage-Strassen multiplication algorithm.

From the above discussion it is clear that the time complexity of our algorithm it heavily depends on the size of the quantities $\beta_1$ and $\beta_2$. As large these quantities are as less practical our attack is. We shall compute bounds for $\beta_1$ and $\beta_2$ in order to evaluate more closely the time complexity of our attack. Using the bounds for $b_{13}$ and $\|\mathbf{b}_1\|$ we obtain $\beta_1 < 3(XY/q)^{1/2}$. In case where $\theta$ is not very close to $90^o$, using the inequalities (1) and $0 < b_{23} < q$, we get $\beta_2 < 1 + C(\theta)(XY/q)^{1/2}$, where $C(\theta) = \sqrt{2}/(\sqrt{3}|cos\theta|)$. We have $\pi/3 \leq \theta \leq 2\pi/3$. Furthermore, suppose that $\theta$ is not between $89.999^o$ and $90.001^o$. It follows that $C(\theta) < 46785$. Hence $\beta_2 < 46785(XY/q)^{1/2}$. Therefore, in case where the keys $a$ and $k$ are quite small, say $[a]_q[k]_q < q(\log q)^4$ and $\theta$ is not very close to $90^o$, we can efficiently compute $a$. Note that the assumption about the angle $\theta$ is not really essential (see Appendix).

It is worthwhile to remark that the step 5 of our algorithm can be parallelized and so its running time can be reduced.

## 4 The DSA ATTACK-2 Algorithm

Put $A = -h(m)s^{-1} \bmod q$ and $B = -rs^{-1} \bmod q$. Then $([a^{-1}]_q, [a^{-1}k]_q)$ is a solution to the congruence $y + Ax + B \equiv 0 \pmod q$. So we can develop an algorithm similar to the previous one, which computes the secret key $a$ in case where $|[a^{-1}]_q| < X$, $|[a^{-1}k]_q| < Y$. The only differences are in Step 5. More precisely, in Steps 5(b) and 5(c) instead of $x_0$ we set $x_0^{-1} \bmod q$.

## 5 The DSA-ATTACK-3 Algorithm

We set $A = h(m)r^{-1} \bmod q$ and $B = -sr^{-1} \bmod q$. Then, $([k^{-1}]_q, [k^{-1}a]_q)$ is a solution to the congruence $y + Ax + B \equiv 0 \pmod q$. Thus, we can develop

an algorithm, similar to the first one, which computes the secret key $a$ in case where $|[k^{-1}]_q| < X$ and $|[k^{-1}a]_q| < Y$. The only differences are in Step 5. More precisely, in Step 5(a) we have not only to compute $x_0$ but also $y_0 = -Ax_0 - B \bmod q$, and in Steps 5(b) and 5(c) instead of $x_0$ (of the first attack) we consider the value $y_0 x_0^{-1} \bmod q$.

## 6 The DSA-ATTACK-4 Algorithm

In this section we give a version of DSA-Attack 1 which the computation of its complexity in terms of $q$ does not involve the angle between the vectors of the basis that we use.

DSA-ATTACK-4
*Input:* One signed message $(m, r, s)$ and integers $X$, $Y$ with $1 < X, Y < q$.

1. Compute $A = \lceil (\sqrt{12qXY - 9Y^2})/3X \rceil$. Put $\mathbf{a} = (AX, Y)$.
2. Compute $\eta \in \{1, \ldots, q-1\}$ such that $-\eta s^{-1} r = A \bmod q$ and next compute $B = -\eta s^{-1} h(m) \bmod q$.
3. Compute, using Lagrange's algorithm, a vector $\mathbf{b} = (b_1 X, b_2 Y)$ having the smallest size among the nonzero vectors of the lattice spanned by $(qX, 0)$ and $\mathbf{a}$.
4. Compute $b_3 = b_2 B \bmod q$.
5. Compute the quantities

$$\alpha = \left\lfloor \frac{B + \sqrt{2}\ \|\mathbf{a}\|}{q} \right\rfloor, \quad \beta = \left\lfloor \frac{b_3 + \sqrt{2}\ \|\mathbf{b}\|}{q} \right\rfloor.$$

6. For every $c \in \{0, \pm 1, \ldots, \pm\alpha\}$ and $d \in \{0, \pm 1, \ldots, \pm\beta\}$,
   (a) compute $x_0 = cb_2 - d + (b_3 - Bb_2)/q$,
   (b) compute $g^{x_0} \bmod p$ (resp. $x_0 P$).
   (c) If $g^{x_0} = R \bmod p$ (resp. $x_0 P = Q$), then output $a = x_0 \bmod q$ and stop. Otherwise output "Failure".

*Proof of Correctness.* Since $k - s^{-1}ra - \eta s^{-1} h(m) \equiv 0 \pmod q$, setting $l = \eta k \bmod q$ and $f(x, y) = y + Ax + B$, we have that $([a]_q, [l]_q)$ is a solution of $f(x, y) \equiv 0 \pmod q$. We shall prove that this algorithm computes correctly $a$ in case where $|[a]_q| \leq X$ and $|[l]_q| \leq Y$.

Denote by $\mathcal{L}$ the lattice spanned by the linearly independent vectors $\mathbf{a}$ and $(qX, 0)$. By Theorem 1, Lagrange's algorithm gives a vector $\mathbf{b} = (b_1 X, b_2 Y)$ of $\mathcal{L}$ with size

$$\|\mathbf{b}\| \leq \sqrt{4/3}\ \det \mathcal{L}^{1/2} \leq \sqrt{4/3}\ (qXY)^{1/2}.$$

Further, we have $\|\mathbf{a}\| > \sqrt{4/3}(qXY)^{1/2}$ and so, we get $\|\mathbf{b}\| < \|\mathbf{a}\|$. If $\mathbf{a}$ and $\mathbf{b}$ are linear dependent, then $b_1 = b_2 A$, whence $\mathbf{b} = b_2 \mathbf{a}$. Hence $\|\mathbf{b}\| \geq \|\mathbf{a}\|$ which is a contradiction. Thus, $\mathbf{a}$ and $\mathbf{b}$ are linearly independent.

Let $\mathbf{b} = \lambda\mathbf{a} + \mu(qX, 0)$, where $\lambda, \mu \in \mathbb{Z}$. Then $\lambda = b_2$. By the Euclidean division of $\lambda B$ by $q$, we have $\lambda B = \pi q + b_3$, where $\pi, b_3 \in \mathbb{Z}$ and $0 \leq b_3 < q$. Put $h(x, y) = b_1 x + b_2 y + b_3$. Therefore

$$h(x, y) \equiv \lambda f(x, y) \pmod{q}.$$

If $q|\lambda$, then we deduce $\|\mathbf{b}\| \geq qY$ and so, $qY < 2(qXY)^{1/2}$, whence $qY < 2X$. Since $Y > 1$ and $X < q$ we have a contradiction. Therefore $q \not| \lambda$ and so, $h(x, y)$ is not identical to zero over $\mathbb{F}_q$. Furthermore, we have $h([a]_q, [l]_q) \equiv 0 \pmod{q}$. On the other hand, we deduce

$$|f([a]_q, [l]_q)| \leq B + \sqrt{2}\,\|\mathbf{a}\|, \quad |h([a]_q, [l]_q)| \leq b_3 + \sqrt{2}\,\|\mathbf{b}\|.$$

Since $h([a]_q, [l]_q) \equiv f([a]_q, [l]_q) \equiv 0 \pmod{q}$, setting

$$\alpha = \left\lfloor \frac{B + \sqrt{2}\,\|\mathbf{a}\|}{q} \right\rfloor, \quad \beta = \left\lfloor \frac{|b_3| + \sqrt{2}\,\|\mathbf{b}\|}{q} \right\rfloor,$$

we get $f([a]_q, [l]_q) = cq$ and $h([a]_q, [l]_q) = dq$, where $c \in \{0, \pm 1, \ldots, \pm\alpha\}$ and $d \in \{0, \pm 1, \ldots, \pm\beta\}$. Since $\mathbf{a}$ and $\mathbf{b}$ are linear independent, we have that the polynomials $f(x, y) - cq$ and $h(x, y) - dq$ are independent and so we obtain $[a]_q = cb_2 - d + (b_3 - Bb_2)/q$.

*Time Complexity.* The Step 1 requires $O((\log qXY)^2)$ and Step 2, $O((\log q)^2)$ bit operations. By Theorem 1, the time complexity of Step 3 is $O((\log qXY)^2)$ bit operations. The use of Euclidean division in Step 4 needs $O((\log q)^2)$ bit operations. Since $\|\mathbf{a}\| \leq 2(qXY)^{1/2}$ and $\|\mathbf{b}\| < \sqrt{4/3}\ (qXY)^{1/2}$, Step 5 needs $O((\log qXY)^2)$ bit operations. Furthermore, we deduce $\alpha, \beta < 3(XY/q)^{1/2}$. So, Step 6, in case of DSA, requires $O((XY/q)(\log p)^2 \log(pXY/q))$ bit operations and, in case of ECDSA, $O((XY/q) \log q \log(XY))$ bit operations and $O(XY(\log q)/q)$ elliptic curve group operations. Therefore, for DSA, the time complexity of our algorithm is $O((XY/q) \log p \log(pXY/q))$ bit operations, and for ECDSA, $O((XY/q) \log q \log(XY))$ bit operations and $O(XY(\log q)/q)$ elliptic curve group operations. Note that when $XY < q(\log q)^4$ our attack is practical. Moreover, the computation of the complexity in terms of $q$ does not use the angle between the vectors $\mathbf{a}$ and $\mathbf{b}$, as in the previous section.

## 7 The DSA-ATTACK-5 Algorithm

Set $B = \lceil (\sqrt{12qXY - 9Y^2})/3X \rceil$. We determine $\eta \in \{1, \ldots, q-1\}$ such that $-\eta s^{-1} h(m_1) \equiv B \pmod{q}$ and next we compute $A = -\eta s^{-1} r \bmod q$. Then taking $l = [\eta k]_q$, we have $l + B[a^{-1}]_q + A \equiv 0 \pmod{q}$. Thus, we can develop an algorithm similar of the above for the computation of $[a^{-1}]_q$ and hence $a$.

## 8 Heuristic Attacks

Every loop of Lagrange's algorithm provide us with linear polynomials $h_1(x, y)$ and $h_2(x, y)$. Taking advantage of this fact we develop an heuristic attack. Put

$A = -rs^{-1} \bmod q$ and $B = -s^{-1}h(m) \bmod q$ and fix bounds $M_1$ and $M_2$. We apply Lagrange's algorithm on the lattice $\mathcal{L}$ spanned by $\mathbf{v}_1 = (A, 1)$ and $\mathbf{v}_2 = (q, 0)$. In each loop the algorithm provide us with a basis $\mathbf{b}_1 = (b_{11}, b_{12})$ and $\mathbf{b}_2 = (b_{21}, b_{22})$ of $\mathcal{L}$. Further, we compute $b_{i3} = b_{i2}B \bmod q$. For every $c_i \in \{0, \pm 1, \ldots, \pm M_i\}$ $(i = 1, 2)$ we compute

$$x_0 = c_1 b_{22} - c_2 b_{12} + \frac{b_{23}b_{12} - b_{13}b_{22}}{q},$$

and next $g^{x_0} \bmod p$ (resp. $x_0 P$). If $g^{x_0} = R \bmod p$ (resp. $x_0 P = Q$), then output $a = x_0 \bmod q$. Similar attacks hold for the other two cases.

## 9 Attacks Using Some Known Bits

Suppose now, as in [10,17,18,3], that some side channel attack or another attack has provided us with some bits of the two keys which consist of some blocks of non-contiguous bits. Then the unknown bits are in, say $d$ blocks, of contiguous bits. So $a = u + a'$ where $a' = \sum_{j=1}^{d} 2^{\lambda_j} a_j$, with $u, \lambda_j$ are known and $a_j$ $(j > 0)$ are unknown. Similarly, $k = v + k'$, where $v$ and $k'$ are positive integers. Then $(a', k')$ is a solution of $y + Ax + B' \equiv 0 \pmod q$, where $A = -rs^{-1} \bmod q$ and $B' = -h(m)s^{-1} + Au + v \bmod q$. If $[a]_q[k]_q < q(\log q)^4$, then we can use the algorithm DSA-ATTACK-1 in order to determine efficiently $a'$ and so $a$. Similarly, we proceed for the other cases.

## 10 Attacks Using Two Signed Messages

Suppose now that we possess two signed messages $(m_1, r_1, s_1)$, $(m_2, r_2, s_2)$ signed with ephemeral keys $k_1$ and $k_2$ respectively. Eliminating $a$ from the resulting congruences, we obtain a new congruence in $k_1$ and $k_2$, and so we can use the above attacks in order to compute $k_1$ or $k_2$ and hence $a$.

## 11 An Example

We consider the elliptic curve $E$ given in [2, Example 3, p. 182] defined over the finite field $\mathbb{F}_p$, where $p = 2^{160} + 7$ is a prime, by the equation

$$y^2 = x^3 + 10x + 1343632762150092499701637438970764818528075565078.$$

The number of points of $E(\mathbb{F}_p)$ is the 160-bit prime

$$q = 1461501637330902918203683518218126812711137002561.$$

Consider the point $P = (x(P), y(P))$ of $E(\mathbb{F}_p)$ of order $q$, where

$$x(P) = 858713481053070278779168032920613680360047535271,$$

$$y(P) = 364938321350392265038182051503279726748224184066.$$

We take as private key the 159− bit integer

$$a = 6327906249263275660959301097845095640888687199 25$$

and so, the public key is $Q = aP = (x(Q), y(Q))$ where

$$x(Q) = 323143541338258768513217516737351163416580216802,$$

$$y(Q) = 108256574209433569821431121048146863357122767205.$$

Let $m$ be a message with hash value

$$h(m) = 142942603167529132165819177831617629501580942.$$

We shall produce two different signatures to $m$ and we shall apply our attacks on the signed messages.

(1) We choose the 160−bit integer

$$k = 137944695433858691899348397673489899972287787 3502$$

as an ephemeral key and we produce the signature $(r, s)$ for $m$, where

$$r = 106878378126871326719764647415952705392498701648 5,$$

$$s = 137856098184546901658844078073912628103985191590 3.$$

We have

$$a^{-1} \bmod \ q = 18014398509481986 < \sqrt{q}\log q,$$

$$a^{-1}k \bmod \ q = 43459871673287390856765008 < \sqrt{q}\log q.$$

We apply DSA-ATTACK-2 with $X = Y = \lfloor \sqrt{q}\log q \rfloor$ and, since $\beta_1 = 141,\ \beta_2 = 176$, we have to check 99899 values.

(2) Next, we choose the 147−bit integer

$$k = 166512230815695319456800029769107765911320439$$

as an ephemeral key and we produce the signature $(r, s)$ for $m$, where

$$s = 130619604428398696689719617650088722834899458 2394,$$

$$r = 724139290201395716568648544869144698056699972161.$$

We have $|[a]_q|, |[k]_q| > q^{0.9}$. Also, $a^{-1} \bmod \ q < \sqrt{q}\log q$ and $|[a^{-1}k]_q| > q^{0.9}$. Further, $|[k^{-1}]_q|$ and $|[k^{-1}a]_q|$ are $> q^{0.9}$ and so, the attacks of Section 3, 4 and 5 do not work. We try the heuristic attacks of Section 3.4. We set $A = -h(m)s^{-1} \bmod q$, $B = -rs^{-1} \bmod q$ and we take $M = 292820000 \simeq 2(\log q)^4$. In 12th loop we obtain $\mathbf{b}_1 = (b_{11}, b_{12})$ and $\mathbf{b}_2 = (b_{21}, b_{22})$, where

$$b_{11} = -100818561570428157294605379284231893689 02, \quad b_{12} = -84226299,$$

$$b_{21} = 101267039418365772399945678097491230654 1, \quad b_{22} = -136503451,$$

which form a basis of the lattice spanned by $(q, 0)$, $(A, 1)$. We compute

$$Bb_{12} \bmod q = -781982649082028777596295134710513450423832598231$$

$$Bb_{22} \bmod q = -206354035197723594446285891701235586069110289872.$$

which are $b_{13}$ and $b_{23}$, respectively. For $c_1 = -197283790$ and $c_2 = -105851969$ we obtain $a^{-1} \bmod q$ and hence $a$.

The above attack can be easily parallelized. We divide the interval $[-M, M]$ to $\kappa = 1000 \approx \lceil (\log q)^{2.54} \rceil$ subintervals of length about $\mu = \lceil 2M/\kappa \rceil = 312745$. For $\varrho = 0, 1, 2, ..., \kappa - 1$, we consider the intervals $I_\varrho = [-M + \varrho\mu, -M + (\varrho+1)\mu]$ and we compute $x_0$ for each integer pair $(c_1, c_2) \in I_i \times I_j$. Thus, this step can be executed in $O(\mu^2)$ steps. For this parallel computation we need $\kappa^2 = 10^6$ processors. Note that the bitcoin grid uses at least $10^{5.2}$ processors and a modern supercomputer uses over $10^5$ processors.

All the computations are executed in a computer with Pentium 2GHz and memory 3Gb. Further for the elliptic curve computations we have used SAGE [20], for the arithmetic computations MAPLE 12 and for the choice of $h(m)$ and $k$ the B.B.S. pseudo-random generator [4]. The secret key $a$ is not chosen at random but in order to support a case where our attack works.

## 12 Conclusion

In this paper we have presented rigorous attacks on the DSA and ECDSA based on Lagrange's algorithm, which recover the secret key in polynomial time, provided that the sizes of the numbers in one of the pairs $([a]_q, [k]_q)$, $([k^{-1}]_q, [k^{-1}a]_q)$ and $([a^{-1}]_q, [a^{-1}k]_q)$ are sufficiently small. Thus, a signer has to take care about the size of all the above quantities and not only of $a$ and $k$. We have also presented an heuristic variant of this attack using all the loops of Lagrange's algorithm and attacks in case where two signed messages are known. An interesting feature of all these attacks are that can be parallelized. If we know a sufficient number of bits of $a$ and $k$, then our attacks can compute $a$. Note that the needed number of known bits is fewer than in [3]. Our attacks can also be applied on other signature schemes where the secret and the ephemeral keys are solutions of a modular bivariate linear equation. Such schemes are Schnorr' signature, Heyst-Pedersen signature, GPS, etc [7,14,21].

## References

1. Bellare, M., Goldwasser, S., Micciancio, D.: "Pseudo-random" number generation within cryptographic algorithms: The DSS case. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 277–291. Springer, Heidelberg (1997)
2. Blake, I.F., Seroussi, G., Smart, N.: Elliptic Curves in Cryptography. Cambridge University Press (2000)

3. Blake, I.F., Garefalakis, T.: On the security of the digital signature algorithm. Des. Codes Cryptogr. 26(1-3), 87–96 (2002)
4. Blum, L., Blum, M., Shub, M.: A Simple Unpredictable Pseudo-Random Number Generator. SIAM Journal on Computing 15, 364–383 (1986)
5. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithm. IEEE Transactions on Information Theory 31, 469–472 (1985)
6. FIPS PUB 186-3: Federal Information Processing Standards Publication, Digital Signature Standard (DSS)
7. Girault, M., Poupard, G., Stern, J.: Global Payment System (GPS): un protocole de signature à la volée. In: Proceedings of Trusting Electronic Trade (1999)
8. Johnson, D., Menezes, A.J., Vanstone, S.A.: The elliptic curve digital signature algorithm (ECDSA). Intern. J. of Information Security 1, 36–63 (2001)
9. Hoffstein, J., Pipher, J., Silverman, J.: An Introduction to Mathematical Cryptography. Springer (2008)
10. Howgrave-Graham, N.A., Smart, N.P.: Lattice Attacks on Digital Signature Schemes. Des. Codes Cryptogr. 23, 283–290 (2001)
11. Koblitz, N., Menezes, A.J., Vanstone, S.A.: The state of elliptic curve cryptography. Des. Codes Cryptogr. 19, 173–193 (2000)
12. Koblitz, N., Menezes, A.J.: A survey of Public-Key Cryptosystems. SIAM Review 46(4), 599–634 (2004)
13. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. 261, 513–534 (1982)
14. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
15. Nguyen, P.Q., Stehlé, D.: Low-Dimensional Lattice Basis Reduction Revisited. ACM Transactions on Algorithms 5(4), Article 46 (2009)
16. Nguyên, P.Q., Stern, J.: The Two Faces of Lattices in Cryptology. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 146–180. Springer, Heidelberg (2001)
17. Nguyen, P.Q., Shparlinski, I.E.: The Insecurity of the Digital Signature Algorithm with Partially Known Nonces. J. Cryptology 15, 151–176 (2002)
18. Nguyen, P.Q., Shparlinski, I.E.: The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces. Des. Codes Cryptogr. 30, 201–217 (2003)
19. Poulakis, D.: Some Lattice Attacks on DSA and ECDSA Applicable Algebra in Engineering. Communication and Computing 22, 347–358 (2011)
20. Stein, W.A., et al.: Sage Mathematics Software (Version 4.6), The Sage Development Team, `http://www.sagemath.org`
21. Stinson, D.R.: Cryptography, Theory and Practice, 2nd edn. Chapman & Hall/CRC (2002)

## Appendix

Here we give some experiential results for the angle $\theta$ between the two reduced vectors of a lattice of the form $L_q = \{(qX, 0), (AX, Y)\}$ after we applied Lagrange algorithm. We considered five random choices for the prime $q$ of 160-bits. For each prime $q$ we have chosen randomly 200 triples of $X, Y, A$ in the interval $(\sqrt{q}, q)$, and then we applied Lagrange algorithm to the lattice $L_q$. In all cases, the angle between the two reduced vectors was not in the interval $I = [89.999^o, 90.001^o]$.

Now we fix $(q, A, B)$ and we let $X, Y$ to vary under the constraints $X > \sqrt{q}$ and $XY < q(\log q)^4$. In each row of the matrix below, we chose a random triple $(q, A, B)$ with $q$ having 160 bits and $A, B$ random integers in the interval $(\sqrt{q}, q)$. We chose 200 random values of $(X, Y)$ under the previous constraints and for each instance $(X, Y)$ we executed Lagrange algorithm. Then we compute the pair $(X, Y)$ which give us the closest angle to $90^o$ (of the output vectors). Finally, for the specific value of $(X, Y)$ we compare the real bound $\beta_2$ with the theoretical bound $C(\theta) \cdot (\log q)^2 \simeq C(\theta) \cdot 160^2$. We summarize the results in the matrix below. We observe that the bound $\beta_2$ is far smaller than the theoretical bound $C(\theta) \cdot (\log q)^2$. This suggests that the algorithm has complexity far smaller than the theoretical given under the assumptions that the angle of the output vectors must be in the interval $I$. Also, if for some choice of $X, Y$ we got an angle in the interval $I$, then a new choice is very likely to give a new anlge not in $I$.

| $(q, A, B)$ | $\beta_2$ | angle $\theta$ | $C(\theta) \cdot (\log q)^2$ |
|---|---|---|---|
| 1 | 14375 | $90.04^o$ | $1170 \cdot 160^2 = 259531 \cdot 10^2$ |
| 2 | 15460 | $89.99^o$ | $4678 \cdot 160^2 = 1197743 \cdot 10^2$ |
| 3 | 64888 | $90.001^o$ | $46782 \cdot 160^2 = 11976330 \cdot 10^2$ |
| 4 | 52999 | $89.984^o$ | $2924 \cdot 160^2 = 748636 \cdot 10^2$ |