Lattice Attacks against Elliptic-Curve Signatures with Blinded Scalar Multiplication

Dahmun Goudarzi, Matthieu Rivain, Damien Vergnaud

Lattice Attacks against Elliptic-Curve Signatures with Blinded Scalar Multiplication

Dahmun Goudarzi[1,2], Matthieu Rivain[1], and Damien Vergnaud[2]

[1] CryptoExperts, Paris, France
dahmun.goudarzi@cryptoexperts.com
matthieu.rivain@cryptoexperts.com

[2] ENS, CNRS, INRIA and PSL Research University, Paris, France
damien.vergnaud@ens.fr

Abstract. Elliptic curve cryptography is today the prevailing approach to get efficient public-key cryptosystems and digital signatures. Most of elliptic curve signature schemes use a *nonce* in the computation of each signature and the knowledge of this nonce is sufficient to fully recover the secret key of the scheme. Even a few bits of the nonce over several signatures allow a complete break of the scheme by lattice-based attacks. Several works have investigated how to efficiently apply such attacks when partial information on the nonce can be recovered through side-channel attacks. However, these attacks usually target unprotected implementation and/or make ideal assumptions on the recovered information, and it is not clear how they would perform in a scenario where common countermeasures are included and where only noisy information leaks via side channels. In this paper, we close this gap by applying such attack techniques against elliptic-curve signature implementations based on a blinded scalar multiplication. Specifically, we extend the famous Howgrave-Graham and Smart lattice attack when the nonces are blinded by the addition of a random multiple of the elliptic-curve group order or by a random Euclidean splitting. We then assume that noisy information on the blinded nonce can be obtained through a template attack targeting the underlying scalar multiplication and we show how to characterize the obtained likelihood scores under a realistic leakage assumption. To deal with this scenario, we introduce a filtering method which given a set of signatures and associated likelihood scores maximizes the success probability of the lattice attack. Our approach is backed up with attack simulation results for several signal-to-noise ratio of the exploited leakage.

1 Introduction

In 1985, Koblitz [Kob87] and Miller [Mil86] independently proposed to use the algebraic structure of elliptic curves in public-key cryptography. Elliptic curve cryptography requires smaller keys and it achieves faster computation and memory, energy and bandwidth savings. It is therefore well suited for embedded devices. There exists several digital signature schemes based on the *discrete logarithm problem* in the group of points of an elliptic curve (e.g. [ElG84,Sch91,Nat00]). These schemes use a *nonce* k for each signed message and compute $[k]\boldsymbol{P}$ for some public point $\boldsymbol{P}$ on the elliptic curve. It is well known that the knowledge of partial information on the nonces used for the generation of several signatures may lead to a total break of the scheme [HS01,Ble00].

Side-channel attacks are a major threat against implementations of cryptographic algorithms [Koc96,KJJ99]. These attacks consists in analyzing the physical leakage of a cryptographic hardware device, such as its power consumption or its electromagnetic emanations. Elliptic curves implementations have been subject to various side-channel attacks. In order to prevent the leakage of partial information on the nonce k from the run of the algorithm that computes scalar multiplication $[k]\boldsymbol{P}$, many countermeasures have been proposed. To thwart simple side-channel analysis, it is customary to ensure a constant (or secret-independent) operation flow (see for instance [Cor99,JY03,IMT02]). To prevent more complex attacks it is necessary to use a probabilistic algorithm to encode the sensitive values such that the cryptographic operations only occur on randomized data. In [Cor99], Coron proposed notably to randomize the scalar k and the projective coordinates of the point $\boldsymbol{P}$. These countermeasures are nowadays widely used and it is not very realistic to assume that a specific set of bits from the nonces could be recovered in clear by a side-channel attacker.

Related works. Two famous attacks have been designed against elliptic-curve signature schemes that exploit partial information on the nonces of several signatures: the Bleichenbacher's attack [Ble00] and the Howgrave-Graham and Smart's attack [HS01]. Nguyen and Shparlinski [NS02,NS03] proposed a proven variant of Howgrave-Graham and Smart's attack when a single block of consecutive bits is unknown to the adversary. Very few results on the security of elliptic-curve signatures with noisy partial information on the nonces are known. In [LPS04], Leadbitter, Page and Smart considered adversaries that can determine some relation amongst the bits of the secret nonces rather than their specific values (but this relation is known with certainty). This work was recently extended by Faugère, Goyet and Renault in [FGR13]. In [BV15], Bauer and Vergnaud designed an attack where the adversary learns partial information on the nonces but not with perfect certainty (but their attack does not apply to ECDSA or Schnorr signatures).

In [CRR03], Chari, Rao and Rohatgi introduced the so-called *template attacks* which aim at exploiting all the available side-channel information when the adversary can only obtain a limited number of leakage traces (which is the case in our discrete logarithm setting since a nonce is used only once). Template attacks require that the adversary is able to perform a profiling of the side-channel leakage (*e.g.* based on a copy of the target device under her control). Template attacks against ECDSA were proposed in [MO09,HMHW09,HM09,MHMP13] but none of them considered a blinded implementation of the scalar multiplication.

Our contributions. We consider *practical* attack scenario, where the target implementation is protected with usual countermeasures and where the adversary recovers some noisy information from a signature computation. We consider an elliptic curve signature based on a regular scalar multiplication algorithm which is protected using classic randomization techniques, such as the masking of projective coordinates and the scalar blinding [Cor99,CJ03]. Our contributions are three-fold.

Firstly, we adapt the lattice-based attack proposed by Howgrave-Graham and Smart [HS01] to the setting where the adversary gets partial information on *blinded nonces* of the form $k + r \cdot q$ where r is a small random integer (typically of 32 bits) and q is the elliptic curve group order [Cor99]. We show that the attack works essentially in the same way than the original one but the number of known bits per nonce must be increased by the bit-length of the random r and the number of unknown blocks of consecutive bits.

Afterwards, we consider a scenario where some noisy information is leaked on the bits of the blinded nonces. Under a realistic leakage assumption, the widely admitted *multivariate Gaussian assumption*, we show how to model the information recovered by a template attacker. Specifically, we characterize the distribution of the obtained likelihood scores with respect to a *multivariate signal-to-noise ratio* parameter. We then introduce a filtering method which, given a set of signatures and associated likelihood scores, select the blinded nonce bits to construct the lattice a way to maximize the success probability of the attack. The method relies on a criteria derived from the analysis of the Howgrave-Graham and Smart's attack and techniques from *dynamic programming*.

Finally, we consider a second implementation setting in which the scalar multiplication is protected by the random Euclidean splitting method [CJ03]. In this setting, the nonces k are split as $k = \lceil k/r \rceil \cdot r + (k \bmod r)$ for a (small) random r and the adversary gets partial information on r, $\lceil k/r \rceil$ and $(k \bmod r)$. We show how this partial information can be use to directly get likelihood scores on the nonce bits and we adapt our filtering method to this scenario. For both blinding schemes, we provide some experimental results for our attack based on several values for the multivariate SNR parameter. Our experiments are simulation-based but one could equally use practically-obtained score vectors from a template attack against an actual implementation. The obtained results would be the same for similar multivariate signal-to-noise ratios.

2 Implementation and Leakage Model

2.1 ECDSA Signature Scheme

The ECDSA signature scheme [Nat00] relies on an elliptic curve E defined over some finite field $\mathbb{K}$ where $E(\mathbb{K})$, the group of $\mathbb{K}$-rational points of E, has (almost) prime order q. The public parameters of the scheme include a description of $E(\mathbb{K})$, a base point $\boldsymbol{P} \in E(\mathbb{K})$ that is a generator of the group (or of the large prime-order subgroup), and a cryptographic hash function H :

$\{0,1\}^* \mapsto [\![0, 2^{\ell-1}]\!]$, where $\ell := \lceil \log_2 q \rceil$. The secret key x is randomly sampled over $[\![0, q-1]\!] = [0, q) \cap \mathbb{Z}$ and the corresponding public key is set as the point $\boldsymbol{Q} = [x]\boldsymbol{P}$, that is the scalar multiplication of $\boldsymbol{P}$ by x.

A signature $\sigma = (t, s)$ of a message $m \in \{0,1\}^*$ is then computed from the secret key x as $t = \mathrm{xcoord}([k]\boldsymbol{P})$ and $s = k^{-1}(h + t \cdot x) \bmod q$, where k is a random nonce sampled over $[\![1, q-1]\!]$ and $h = \mathrm{H}(m)$. One can then verify the signature from the public key $\boldsymbol{Q}$ by computing $u = s^{-1}\, h \bmod q$ and $v = s^{-1}\, t \bmod q$, and checking whether $\mathrm{xcoord}([u]\boldsymbol{P} + [v]\boldsymbol{Q}) = t$. A legitimate signature indeed satisfies $[u]\boldsymbol{P} + [v]\boldsymbol{Q} = [s^{-1}(h + t \cdot x) \cdot h \bmod q]\boldsymbol{P} = [k]\boldsymbol{P}$.

2.2 Target Implementation

The attacks presented in this paper target an ECC-signature implementation that relies on a regular scalar multiplication algorithm, which is assumed to be binary in the sense that each loop iteration handles a single bit of the scalar. A prominent example of such a binary regular algorithm is the Montgomery ladder [Mon87] which is widely used for its security and efficiency features [JY03,IMT02,GJM+11].

The target implementation is also assumed to include common countermeasures against side-channel attacks such as as the classic scalar blinding [Cor99] and the Euclidean blinding [CJ03]:

Classic blinding scheme:

1. $r \overset{\$}{\leftarrow} [\![0, 2^\lambda - 1]\!]$
2. $a \leftarrow k + r \cdot q$
3. return $[a]\boldsymbol{P}$

Euclidean blinding scheme:

1. $r \overset{\$}{\leftarrow} [\![1, 2^\lambda - 1]\!]$
2. $a \leftarrow \lfloor k/r \rfloor$; $b \leftarrow k \bmod r$
3. return $[r]([a]\boldsymbol{P}) + [b]\boldsymbol{P}$

2.3 Leakage Model

The computation performed during one iteration of any regular binary scalar multiplication is deterministic with respect to the current scalar-bit b and the two points $\boldsymbol{P}_0$ and $\boldsymbol{P}_1$ in input of the iteration. The side-channel leakage produced in such an iteration can hence be modeled as a noisy function $\psi(b, \boldsymbol{P}_0, \boldsymbol{P}_1)$. In the following we shall assume that for randomized points $(\boldsymbol{P}_0, \boldsymbol{P}_1)$, the leakage $\psi(b, \boldsymbol{P}_0, \boldsymbol{P}_1)$ can be modeled by a multivariate Gaussian distribution:

$$\psi(b, \boldsymbol{P}_0, \boldsymbol{P}_1) \sim \mathcal{N}(m_b, \Sigma) \ , \tag{1}$$

where m_0 and m_1 are T-dimensional mean leakage vectors and where Σ is a $T \times T$ covariance matrix. In what follows, we shall simply denote this leakage $\psi(b)$. The overall leakage of the scalar multiplication $[a]\boldsymbol{P}$ is hence modeled as $\big(\psi(a_{\ell_a - 1}), \ldots, \psi(a_1), \psi(a_0)\big)$ where we further assume the mutual independence between the $\psi(a_i)$ (where the length ℓ_a of a depends on the randomization scheme).

2.4 Profiling Attack

We consider a profiling attacker that owns templates for the leakage of a scalar multiplication iteration w.r.t. the input bit. Based on these templates, the attacker can mount a maximum likelihood attack to recover each bit of the scalar with a given probability. More precisely, the considered attacker can

- measure the side-channel leakage of a scalar multiplication $[a]\boldsymbol{P}$,
- divide the measured traces into sub-traces $\psi(a_i)$,
- compare the measured leakage with templates for $\psi(0)$ and $\psi(1)$, and determine the probability that $a_i = 0$ or $a_i = 1$ given an observation $\psi(a_i)$.

In particular, we consider the ideal case where the attacker knows the exact distribution of $\psi(0)$ and $\psi(1)$, namely he knows the leakage parameters m_0, m_1 and Σ. Although this might be viewed as a strong assumption, efficient techniques exist to derive precise leakage templates in practice, especially in our context where the key-space is of size 2 ($b \in \{0,1\}$). Even if model-error might lower the probability of correctly recovering a target bit in practice, it would not invalidate the principle of our attacks.

Considering the above leakage model, the probability that the bit a_i equals $b \in \{0,1\}$ given a leakage sample $\psi(a_i) = x_i$ satisfies

$$p_b(x_i) := \Pr(a_i = b \mid \psi(a_i) = x_i) \propto \exp\big(-\frac{1}{2}(x_i - m_b)^{\mathsf{t}} \cdot \Sigma^{-1} \cdot (x_i - m_b)\big) \ , \quad (2)$$

where $\propto$ means *is equal up to a constant factor*, such that $p_0(x_i) + p_1(x_i) = 1$. Let us define the *multivariate signal-to-noise ratio* (SNR) θ as

$$\theta = \Lambda \cdot (m_0 - m_1) \ , \quad (3)$$

where Λ is the Cholesky decomposition matrix of Σ^{-1}, that is the upper triangular matrix satisfying $\Lambda^{\mathsf{t}} \cdot \Lambda = \Sigma^{-1}$. This decomposition always exists provided that Σ is full-rank (*i.e.* no coordinate variable of the multivariate Gaussian is the exact linear combination of the others) which can be assumed without loss of generality. We have the following result:

Proposition 1. *For every $x_i \in \mathbb{R}^{\mathsf{t}}$*

$$p_0(x_i) \propto \begin{cases} \exp\big(-\frac{1}{2}y_i^{\mathsf{t}}y_i\big) & \text{if } a_i = 0 \\ \exp\big(-\frac{1}{2}(y_i^{\mathsf{t}}y_i + 2\theta^{\mathsf{t}}y_i + \theta^{\mathsf{t}}\theta)\big) & \text{if } a_i = 1 \end{cases} \quad (4)$$

where $y_i = \Lambda \cdot (x_i - m_{a_i})$. Moreover if x_i follows a distribution $\mathcal{N}(m_{a_i}, \Sigma)$, then y_i follows a distribution $\mathcal{N}(\boldsymbol{0}, I_T)$, where I_T is the identity matrix of dimension T.

The above proposition shows that, for any T-dimensional leakage distribution, the outcome of the considered template attack only depends on the multivariate SNR θ. That is why, in Section 6 we provide attack experiments for several values of θ. In practice, one could evaluate the vulnerability of an implementation to our attack by constructing leakage templates for $\psi(0)$ and $\psi(1)$, deriving the corresponding multivariate SNR θ, and simulating probability scores based on Proposition 1.

3 Lattice Attack with Partially-Known Blinded Nonces

In this section, we recall the lattice attack by Howgrave-Graham and Smart [HS01] against ECDSA with partially known nonces, and we extend it to the case where the attacker has partial information on the blinded nonces. The attacker is assumed to collect $n+1$ signatures $\sigma_0, \sigma_1, \ldots, \sigma_n$, for which he knows some bits of the blinded nonces $a_0, a_1, \ldots, a_n$. These blinded nonces are defined as $a_i = k_i + r_i \cdot q$, where k_i is the original (non-blinded) nonce, and r_i is the λ-bit random used in the blinding of k_i. Additionally, we denote by t_i and s_i the two parts of the ECDSA signature $\sigma_i = (t_i, s_i)$ and by h_i the underlying hash value.

3.1 Attack Description

By definition, we have $s_i - a_i^{-1}(h_i + t_i x) \equiv 0 \bmod q$ for every signature σ_i. We can then eliminate the secret key x from the equations since we have

$$x \equiv \frac{a_i s_i - h_i}{t_i} \bmod q \qquad \Longrightarrow \qquad \frac{a_0 s_0 - h_0}{t_0} \equiv \frac{a_i s_i - h_i}{t_i} \bmod q \tag{5}$$

for every $i \in \{1, \ldots, n\}$. The above can then be rewritten as

$$a_i + A_i a_0 + B_i \equiv 0 \bmod q \tag{6}$$

where $A_i = -\frac{s_0 t_i}{s_i t_0} \bmod q$ and $B_i = \frac{h_0 t_i - h_i t_0}{t_0 s_i} \bmod q$.

The goal of the attack is to use the information we have on each a_i to derive a system of equations which can be reduced to a lattice *closest vector problem* (CVP). Let us assume we can obtain several blocks of consecutive bits so that a_i can be expressed as:

$$a_i = \sum_{j=1}^{N} x_{i,j} 2^{\kappa_{i,j}} + x_i' \tag{7}$$

where x_i' is known, and $x_{i,1}, x_{i,2}, \ldots, x_{i,N}$ are the N unknown blocks. We will denote by $\mu_{i,j}$ the bit-length of each $x_{i,j}$ so that we have $0 \le x_{i,j} < 2^{\mu_{i,j}}$.

We can now rewrite (6) to obtain a system of linear equations in the $x_{i,j}$ as follows:

$$x_{i,1} + \sum_{j=2}^{N} \alpha_{i,j} x_{i,j} + \sum_{j=1}^{N} \beta_{i,j} x_{0,j} + \gamma_i = \eta_i q \tag{8}$$

for some known coefficients $\alpha_{i,j}, \beta_{i,j}, \gamma_i \in [\![0, q-1]\!]$, and unknown integers $x_{i,j}$ and η_i. Let $C \in \mathcal{M}_{n,N(n+1)-n}(\mathbb{Z})$ be the matrix defined as

$$C = \begin{pmatrix} \alpha_{1,2} \cdots \alpha_{1,N} & & & & \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,N} \\ & \alpha_{2,2} \cdots \alpha_{2,N} & & & \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,N} \\ & & \ddots & & \vdots & \vdots & \ddots & \vdots \\ & & & \alpha_{n,2} \cdots \alpha_{n,N} & \beta_{n,1} & \beta_{n,2} & \cdots & \beta_{n,N} \end{pmatrix} \tag{9}$$

and let $\boldsymbol{x} \in \mathbb{Z}^{N(n+1)-n}$ and $\boldsymbol{\eta} \in \mathbb{Z}^n$ be the (column) vectors defined as

$$\boldsymbol{x} = (x_{1,2}, \dots, x_{1,N} \mid x_{2,2}, \dots, x_{2,N} \mid \dots \mid x_{n,2}, \dots, x_{n,N} \mid x_{0,1}, \dots, x_{0,N})^{\mathsf{t}} ,$$
$$\boldsymbol{\eta} = (\eta_1, \eta_2, \dots, \eta_n)^{\mathsf{t}} .$$

According to (7), we have

$$C \cdot \boldsymbol{x} - q \cdot \boldsymbol{\eta} = -(\gamma_1 + x_{1,1}, \gamma_2 + x_{2,1}, \ \dots, \ \gamma_n + x_{n,1})^{\mathsf{t}} \tag{10}$$

We consider the lattice $\mathcal{L}$ spanned by the columns of the following matrix:

$$M = \begin{pmatrix} C & -q \cdot I_n \\ -I_{N(n+1)-n} & 0 \end{pmatrix} \tag{11}$$

where I_n and $I_{N(n+1)-n}$ denote the identity matrices of dimension n and $N(n+1)-n$ respectively. In particular, the vector $\boldsymbol{y}^{\mathsf{t}} = -(\boldsymbol{x}^{\mathsf{t}} | \boldsymbol{\eta}^{\mathsf{t}}) \in \mathbb{Z}^{N(n+1)}$ yields the following lattice vector:

$$M \cdot \boldsymbol{y} = (\gamma_1 + x_{1,1}, \ \gamma_2 + x_{2,1}, \ \dots, \ \gamma_n + x_{n,1} \mid \boldsymbol{x}^{\mathsf{t}} \ )^{\mathsf{t}} \tag{12}$$

which might be close to the non-lattice vector

$$\boldsymbol{v} = (\gamma_1, \gamma_2, \dots, \gamma_n, 0, 0, \dots, 0)^{\mathsf{t}} \tag{13}$$

It is indeed easy to check that the Euclidean distance $\|M \cdot \boldsymbol{y} - \boldsymbol{v}\|$ satisfies

$$\|M \cdot \boldsymbol{y} - \boldsymbol{v}\|^2 = \sum_{\substack{0 \le i \le n \\ 1 \le j \le N}} x_{i,j}^2 \le \sum_{\substack{0 \le i \le n \\ 1 \le j \le N}} 2^{2\mu_{i,j}} \tag{14}$$

If the distance is small enough, and if the lattice dimension $(n+1)N$ is not too high, one can find $M \cdot \boldsymbol{y}$ as the closest lattice vector to $\boldsymbol{v}$ and hence get the solution to the system. From the latter solution, one can recover the randomized nonces a_i (by (7)) which in turn yields the secret key x (by (5)).

Normalization. We consider unknown blocks $x_{i,j}$ that may be of different bit-lengths $\mu_{i,j}$. Therefore, we shall normalize the matrix M in order to have the same weight in all the coordinates of $M \cdot \boldsymbol{y}$, which shall lead to a better heuristic on the resolution of the lattice problem. To do so, let us define $\rho_{i,j} \in \mathbb{Z}$ as

$$\rho_{i,j} = \frac{1}{2^{\mu_{i,j}}} . \tag{15}$$

We then consider the lattice $\mathcal{L}'$ spanned by the rows of the matrix $M' = D \cdot M$, where D denotes the following diagonal matrix:

$$D = \mathcal{D}\big((\rho_{i,1})_{i=1}^n \mid (\rho_{1,j})_{j=2}^N \mid (\rho_{n,j})_{j=2}^N \mid (\rho_{0,j})_{j=1}^N\big) \tag{16}$$

where $\mathcal{D}$ is the function mapping a vector to the corresponding diagonal matrix.[3] The non-lattice vector $\boldsymbol{v}'$ is then defined as

$$\boldsymbol{v}' = D \cdot \boldsymbol{v} = (\rho_{1,1}\gamma_1, \rho_{2,1}\gamma_2, \ldots, \rho_{n,1}\gamma_n, 0, 0, \ldots, 0)^{\mathsf{t}}$$

and the target closest lattice vector is $M' \cdot \boldsymbol{y}$. The distance between these two vectors then satisfies

$$\|M' \cdot \boldsymbol{y} - \boldsymbol{v}'\| = \|D \cdot (M \cdot \boldsymbol{y} - \boldsymbol{v})\| = \Big(\sum_{\substack{0 \le i \le n \\ 1 \le j \le N}} (\rho_{i,j} x_{i,j})^2 \Big)^{\frac{1}{2}} \le \sqrt{(n+1)N} \ . \quad (17)$$

3.2 Attack Parameters

We make the classic heuristic assumption that the CVP can be efficiently solved as long as the dimension is not too high and the distance is upper-bounded by

$$\|M' \cdot \boldsymbol{y} - \boldsymbol{v}'\| \le c_0 \sqrt{\dim(M')} \det(M')^{\frac{1}{\dim(M')}} \ , \quad (18)$$

for some constant $c_0 = 1/\sqrt{2\pi e} \simeq 0.24197$ (see for instance [FGR13]). In our attacks, we will actually use the polynomial-time LLL [LLL82] algorithm in order to recover the lattice vector $M' \cdot \boldsymbol{y}$. This gives a slightly worse inequality in (18) and the CVP can be efficiently solved for smaller values c_0 that may actually depend on the dimension of the lattice.

Let us denote by $\mu_i = \sum_j \mu_{i,j}$ the number of unknown bits in a_i, and by $\mu = \sum_i \mu_i$ the total number of unknown bits. Let us also denote by $\delta_i = \ell + \lambda - \mu_i$ the number of known bits in a_i, and by $\delta = \sum_i \delta_i = (n+1)(\ell+\lambda) - \mu$ the total number of known bits. The dimension of the matrix M' is $(n+1)N$ and its determinant is $\det(M') = \det(M) \cdot \det(D) = q^n \cdot 2^{-\mu}$. By (17), the above inequality is satisfied whenever we have[4]

$$1 \le c_0 (q^n \cdot 2^{-\mu})^{\frac{1}{N_b}} \iff \mu + c_1 N_b \le n \cdot \log_2(q) < n\ell \ , \quad (19)$$

where $N_b = (n+1)N$ denotes the total number of unknown blocks in the $n+1$ signatures, and where c_1 is a constant defined as $c_1 = -\log_2(c_0) > 0$. We can deduce a sufficient condition on the number of bits δ that must be known for the attack to succeed:

$$\delta \ge \ell + (n+1)\lambda + c_1 N_b \iff \sum_{i=0}^{n} (\delta_i - \lambda - c_1 N) \ge \ell \ . \quad (20)$$

This means that in order to mount a lattice attack against ECDSA protected with classic randomization of the nonce, one must recover at least $(\lambda + c_1 N)$ bits

[3] In practice, we might take $\rho_{i,j} = 2^{\mu_{max} - \mu_{i,j}}$ where $\mu_{max} = \max_{i,j} \mu_{i,j}$ to work over the integers.

[4] In [HS01], Howgrave-Graham and Smart overlooked the linear dependency in N_b in (19) as they assumed $c_0 = 1$. As we show in Section 3.3, this increase in the number of known bits is not an artifact of the technique but is actually necessary.

of each blinded nonce plus some extra bits, where the total amount of extra bits must reach ℓ.

Varying number of blocks. For the sake of clarity, we described the attack by assuming a constant number N of unknown blocks for all the signatures. Note that the attack works similarly with a varying number of blocks N_0, N_1, ..., N_n and the above condition keep unchanged. To see this, simply consider the above description with $N = \max_i N_i$ and $\mu_{i,j} = 0$ for every $j > N_i$. Note however that for different block sizes, we have $N_b = \sum_{i=0}^{n} N_i$ instead of $N_b = (n+1)N$.

3.3 Experiments

The CVP can be solved using Babai's rounding algorithm [Bab86], but in practice one shall prefer the *embedding technique* from [GGH97]. This technique reduces the CVP to the *shortest vector problem* (SVP), by defining the *embedding lattice* $\mathcal{L}_{\text{emb}}$ spanned by the columns of

$$\left(\begin{array}{c|c} M' & \boldsymbol{v} \\ \hline 0 \cdots 0 & -K \end{array}\right) \tag{21}$$

where K is a constant taken to be of the same order as the greatest coefficient of M'. Informally, if $M' \cdot \boldsymbol{y} \in \mathcal{L}$ is close to $\boldsymbol{v}$, then $\boldsymbol{w} = \big((M' \cdot \boldsymbol{y} - \boldsymbol{v})^{\mathsf{t}} \mid K\big)^{\mathsf{t}}$ is a short vector of the embedding lattice $\mathcal{L}_{\text{emb}}$. By applying LLL to $\mathcal{L}_{\text{emb}}$, one can therefore recover $\boldsymbol{w}$ as the shortest vector in the reduced basis, from which $M' \cdot \boldsymbol{y}$ is easily deduced.

In order to validate the constraint on the parameters (see (20)) and determine the actual value of c_1, we experienced the attack using the embedding technique with different set of parameters. Specifically, we used the ANSSI 256-bit elliptic curve ($\ell = 256$), different random sizes ($\lambda \in \{0, 16, 32, 64\}$), different numbers of signatures ($(n+1) \in \{5, 10, 20\}$), as well as different numbers of blocks per signature ($N \in \{1, 2, 5, 10\}$). In each experiments, the blocks were randomly distributed in the blinded nonces (by randomly sampling the starting index among possible ones). Additionally, we considered that the number of bits per block could vary according to a standard deviation parameter σ as follows. The bit-length $\delta_{i,j}$ of each block is randomly sampled according to the distribution $\mathcal{N}(m, \sigma)$ with $m = \delta/N_b$, where we recall that $\delta = \sum_{i,j} \delta_{i,j}$ is the total number of known bits and $N_b = (n+1)N$ is the total number of unknown blocks. Samples are then rounded to the nearest integer with rejection for samples lower than 1 (minimum number of bits per block). Finally, samples are randomly incremented or decremented until they sum to the desired value δ. In a nutshell, taking a standard deviation $\sigma = 0$ makes all the $\delta_{i,j}$'s equal to m. On the other hand, taking a standard deviation to $\sigma \approx m$ makes the $\delta_{i,j}$'s to vary over $[\![1, 2m]\!]$ (with a few ones beyond $2m$). For our experiments, we took $\sigma = 0.1m$ (slight deviation), $\sigma = 0.5m$ (medium deviation), and $\sigma = m$ (strong deviation).

In each setting, the number of known bits is set to $\delta = \ell + (n+1)\lambda + \tau$ for a varying margin τ that shall correspond to $c_1 N_b$. For each tested value of τ, we record the success rate of the attack over 100 trials. Table 1 summarizes the obtained ratio τ_{95}/N_b where τ_{95} is the margin required to obtain a 95% success rate. This ratio gives a good estimation of the practical value of c_1.

Table 1. Ratio τ_{95}/N_b for various set of parameters.

	$(n+1)=5$				$(n+1)=10$				$(n+1)=20$		
$N_b =$	5	10	25	50	10	20	50	100	20	40	100
$\lambda = 0$ (slight dev)	2.80	2.50	2.36	2.90	3.90	3.55	3.58	4.62	5.20	4.90	5.26
$\lambda = 0$ (medium dev)	3.60	2.60	2.56	2.90	4.10	3.30	3.52	3.57	4.85	4.42	4.51
$\lambda = 0$ (strong dev)	3.20	2.50	2.28	2.54	4.10	2.95	3.14	3.07	5.25	4.17	3.93
$\lambda = 16$ (slight dev)	3.40	2.80	2.44	2.90	3.80	3.35	3.62	4.75	5.05	4.95	5.5
$\lambda = 16$ (medium dev)	3.40	2.60	2.40	3.02	4.20	3.15	3.40	4.20	5.25	4.77	4.96
$\lambda = 16$ (strong dev)	3.60	2.50	2.36	2.66	3.70	3.05	3.24	3.29	5.25	4.67	4.28
$\lambda = 32$ (slight dev)	3.80	2.70	2.68	3.06	3.80	3.45	3.74	4.71	4.75	4.70	5.17
$\lambda = 32$ (medium dev)	3.40	2.60	2.60	2.68	3.90	3.10	3.60	4.07	4.95	4.50	5.12
$\lambda = 32$ (strong dev)	3.00	2.90	2.36	2.60	4.00	3.05	3.32	3.41	4.90	4.73	4.62
$\lambda = 64$ (slight dev)	3.00	2.90	2.52	2.98	3.80	3.35	3.44	4.72	4.70	4.77	5.24
$\lambda = 64$ (medium dev)	3.20	2.80	2.36	2.98	3.70	3.55	3.68	4.31	4.80	4.60	5.23
$\lambda = 64$ (strong dev)	3.20	2.80	2.44	2.72	3.50	3.40	3.68	3.78	5.45	4.30	4.75

We observe that for the tested parameters, the experimental value of c_1 lies between 3 and 5 most of the time. We also observe that the size of the random (λ) and the deviation on the number of bits per known blocks have a small impact on the resulting c_1, whereas the number of signatures and the total number of blocks have a stronger impact.

4 Attacking Implementations with Classic Blinding

In this section we focus on attacking implementations of elliptic-curve signatures that leak side-channel information on the blinded nonce as described in Section 2. In this model, one performs a template attack on the randomized scalar multiplication and recovers a probability score $\Pr(a_{i,j} = 0)$ for every bit $a_{i,j}$ of every blinded nonce a_i. The goal is then to select a set of bits among all the recovered *noisy bits* that has the required properties for solving the associated lattice system and that has the highest possible probability of success (*i.e.* all the selected bits must have been correctly guessed based on the probability scores).

For the lattice construction, we shall use the most probable value $\hat{a}_{i,j}$ of each bit $a_{i,j}$, that is

$$\hat{a}_{i,j} = \underset{b \in \{0,1\}}{\operatorname{argmax}} \Pr(a_{i,j} = b) , \tag{22}$$

and we shall denote by $p_{i,j}$ the probability that the bit $a_{i,j}$ is correctly guessed, that is

$$p_{i,j} = \Pr(\hat{a}_{i,j} = a_{i,j}) = \max_{b \in \{0,1\}} \Pr(a_{i,j} = b) . \tag{23}$$

Let I denote the set of indices corresponding to the selected signatures $\{\sigma_i\}_{i\in I}$ and let J_i denote the set of indices corresponding to the selected guessed bits $\{\hat{a}_{i,j}\}_{i\in J_i}$ within a random nonce for every $i \in I$. Then we will use the bits $\{\hat{a}_{i,j} | i \in I, j \in J_i\}$ to construct the lattice. The CVP's algorithm will only succeed in recovering the right solution if all these bits are correctly guessed, namely if $\hat{a}_{i,j} = a_{i,j}$ for every $i \in I$ and $j \in J_i$. The probability p that these success events happen satisfies

$$p = \prod_{i\in I} p_i \quad \text{with} \quad p_i = \prod_{j\in J_i} p_{i,j} \ , \tag{24}$$

where p_i is the probability that all the guessed bits within the random nonce a_i are correct. Our goal is hence to define the sets I, and J_i for every $i \in I$ such that the success probability is maximal and such that the prerequisites of the lattice attack are well met.

Let N_i denote the number of unknown blocks in the blinded nonce a_i, *i.e.* the number of non-adjacent blocks of indices $j \notin J_i$. The dimension Δ of the lattice satisfies $\Delta = \sum_{i\in I} N_i$ (see Section 4). Let Δ_{max} be the maximal dimension of a lattice for which the attack can be practical, *i.e.* the LLL-reduction can be done in a reasonable time.[5] The selected bits must then satisfy

$$\sum_{i\in I} N_i \leq \Delta_{max} \ . \tag{25}$$

Following Section 4, we denote by $\delta_i = |J_i|$ the number of recovered bits in the nonce a_i. We recall the necessary condition for the lattice reduction to work:

$$\sum_{i\in I} (\delta_i - \lambda - c_1 N_i) \geq \ell \ . \tag{26}$$

From the two above conditions, a requirement for the selected bits within a blinded nonce a_i is to satisfy:

$$\frac{\delta_i - \lambda}{N_i} > \frac{\ell}{\Delta_{max}} + c_1 \ . \tag{27}$$

Otherwise, the contribution in terms of known bits is too small with respect to the increase of the lattice dimension. In other words, if the above condition is not satisfied then the maximal dimension Δ_{max} is reached before the number of necessary known bits.

The above condition on the pair (δ_i, N_i) is not sufficient in itself as it does not specify how to select the sets J_i maximizing the success probability. Our goal is now to define a sound criterion on each signature that will allow us to

[5] For our experiments, we took $\Delta_{max} = 256$ for which *SageMath* (using the fplll library) performs the reduction within 20 seconds on a 2.0 GHz Intel Xeon E5649 processor.

select the best sets J_i *i.e.* the set maximizing the success probability defined in (24). Maximizing this probability is equivalent to maximizing the log-success probability $\log(p) = \sum_{i \in I} \log(p_i)$, and maximizing the latter sum while satisfying (26) can be done by selecting the sets J_i with maximal ratio $\frac{\log p_i}{\delta_i - \lambda - c_1 N_i}$. We hence look for the set of indices J_i that maximizes the following value

$$\gamma_i = p_i^{\frac{1}{\delta_i - \lambda - c_1 N_i}} \ . \tag{28}$$

Let $f \colon N \mapsto \lceil \frac{\ell \cdot N}{\Delta_{max}} \rceil + c_1 N + \lambda$ be the function that define the minimum number of known bits from a number of unknown blocks for condition (27) to hold. For each signature σ_i and for each possible number of blocks $N \in [\![1, N_{max}]\!]$ we define:

$$J_i(N) = \underset{\substack{J; |J| = f(N) \\ g(J) \leq N}}{\operatorname{argmax}} \Big(\prod_{j \in J_i} p_{i,j} \Big) \ , \tag{29}$$

where the function $g(J)$ gives the number of unknown blocks in the set J.

We then define

$$p_i(N) = \prod_{j \in J_i(N)} p_{i,j} \quad \text{and} \quad \gamma_i(N) = p_i(N)^{\frac{1}{f(N) - \lambda - c_1 N}} \ . \tag{30}$$

Eventually, we shall set

$$N_i = \underset{N}{\operatorname{argmax}}\, \gamma_i(N) \ , \quad \gamma_i = \gamma_i(N_i) \ , \quad J_i = J_i(N_i) \ , \quad \text{and} \quad \delta_i = f(N_i) \ .$$

Remark 1. At this point, a natural idea is to take I as the set of indices with the highest values γ_i. However, this strategy is not reliable in practice. Indeed, it can occur that a bad guess $\hat{a}_{i,j} \neq a_{i,j}$ comes with a strong probability $p_{i,j}$ (*i.e.* one gets a strong confidence in the wrong choice). Since each γ_i aggregates many probabilities $p_{i,j}$, the occurrence of such an extreme event is only marginally correlated to the actual value of γ_i.

In view of the above remark, our approach in practice is to try several random combinations of signatures, namely to take I uniformly at random among the subsets of signatures (tightly) satisfying the constraint (26).

Evaluating Equation (29)**.** We now explain how to evaluate the function

$$N \mapsto \max_{\substack{J; |J| = f(N) \\ g(J) \leq N}} \Big(\prod_{j \in J_i} p_{i,j} \Big) \tag{31}$$

for some family of probabilities $p_{i,0}, p_{i,1}, \ldots, p_{i,\ell+\lambda-1}$ (getting the argmax is then straightforward). For this purpose, we define maxp as the function mapping a triplet (j, δ, N) to the above max but with the condition $J \subseteq [\![j, \ell + \lambda - 1]\!]$, $|J| = \delta$, and $g(J) \leq N$, so that the target function in (31) can be rewritten as:

$$N \mapsto \mathsf{maxp}(0, f(N), N) \tag{32}$$

The maxp function can then be evaluated by the following recurrence relation:

$$\mathsf{maxp}(j, \delta, N) = \max\big(\mathsf{block}(j, \delta, N), \mathsf{next}(j, \delta, N)\big) \tag{33}$$

where

$$\mathsf{block}(j, \delta, N) = \max_{w \in [\![1,\delta]\!]} \Big(\prod_{k=j}^{j+w-1} p_{i,k} \Big) \mathsf{maxp}(j + w + 1, \delta - w, N - 1) \ , \tag{34}$$

$$\mathsf{next}(j, \delta, N) = \mathsf{maxp}(j + 1, \delta, N) \ . \tag{35}$$

The term $\mathsf{block}(j, \delta, N)$ in the above max represents the case where the next w bits are added to the set J, while the term $\mathsf{next}(j, \delta, N)$ is for the case where the next bit is not taken in the set J.[6] The tail of the recursion is fourfold:

$$\mathsf{maxp}(j, \delta, N) = \begin{cases} 0 & \text{if } j + \delta > \ell + \lambda \\ \prod_{k=j}^{\ell+\lambda-1} p_{i,k} & \text{if } j + \delta = \ell + \lambda \\ 0 & \text{if } N = 0 \text{ and } \delta > 0 \\ 1 & \text{if } \delta = 0 \end{cases} \tag{36}$$

Remark 2. In order to get a higher success probability, we can also use exhaustive search to guess certain bits in a signature. In fact, between two blocks of bits selected by the block function, we can find a small block of bits with poor probability score which will discard the chance of selecting a larger unique block. To tackle this issue and improve the probability of success, we can use exhaustive search to recover these bits with probability 1 (independently of the underlying $p_{i,j}$ values). This improvement of our method is described in Appendix A.

5 Attacking Implementations with Euclidean Blinding

In this section we focus on attacking implementations protected with Euclidean blinding [CJ03], *i.e.* for which the nonces are decomposed as $k_i = a_i \cdot r_i + b_i$ for $i \in \{1, \ldots, n\}$ where r_i is picked uniformly at random over $[\![1, 2^\lambda - 1]\!]$, $a_i = \lceil \frac{k_i}{r_i} \rceil$ and $b_i = k_i \mod r_i$ (see Section 3). We suppose that the attacker can recover the probability score for every bit of a_i, r_i and b_i for $i \in \{1, \ldots, n\}$ with the same template attack as in the previous section.

One approach to mount an attack is to use the information we have on each nonce to derive a system of equations which we may solve using a lattice attack. The equations are of degree 2 and the number of involved monomials increases quadratically with the number of unknown blocks. A natural idea is to use the famous Coppersmith method [Cop96b,Cop96a] which is a family of lattice-based techniques to find small integer roots of polynomial equations. The method generates more non-linear equations by multiplication of several non-linear equations

[6] A particular case occurs when $j = 0$ for which the recursive call to maxp in block does not decrement N since no unknown block appears before the index $j = 0$.

before the linearization step in order to improve attacks. However, in this case the structure of the variables is complex and even using polynomials of small degree leads to a lattice of very large dimension.

Our approach is different and practical: from the recovered noisy bits, one can compute probability scores for the individual bits of the nonces k_i themselves. If the probability that the individual bits in a_i, r_i and b_i for $i \in [\![1, n]\!]$ are all correctly guessed is equal to $1/2 + \varepsilon$, for some $\varepsilon > 0$, it can be checked that the obtained probabilities $\Pr(\hat{k}_{i,j} = k_{i,j})$ that the j-th bit of the nonce k_i is correctly guessed tends towards $1/2$ quickly as j grows towards the middle bit $\ell/2$.

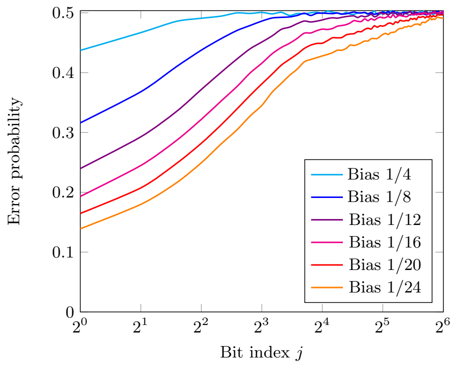

Fig. 1. Estimated error probability of individual nonce bits

To show this fact, we ran the following experiments:

- for several bias $\varepsilon \in \{1/4, 1/8, \ldots, 1/24\}$, we picked uniformly at random 10,000 nonces k_i and random r_i (for $\lambda = 16$) and we computed the corresponding a_i and b_i such that $a_i \cdot r_i + b_i = k_i$;
- we computed noisy versions of $(\tilde{a}_i, \tilde{r}_i, \tilde{b}_i)$ of (a_i, r_i, b_i) as if they were transmitted over a binary symmetric channel with crossover probability ε and we computed $\tilde{k}_i = \tilde{a}_i \cdot \tilde{r}_i + \tilde{b}_i$;
- for each bias and each individual bit, we computed the proportion of $\tilde{k}_{i,j}$ that matches $k_{i,j}$.

The estimated error probability $\Pr(\tilde{k}_{i,j} \neq k_{i,j})$ is plotted in Figure 1 for the first 64 bits (in log-scale for the x-axis). These experiments show that the probabilities $\Pr(\hat{k}_{i,j} = k_{i,j})$ tends towards $1/2$ exponentially as j comes close to the middle bit $\ell/2$. For this reason, in the case of the Euclidean blinding, we will only focus on two particular blocks of k_i for each signature: the block of $\delta_{i,1}$ least significant bits (lsb) and the block of $\delta_{i,2}$ most significant bits (msb), for some $\delta_{i,1}, \delta_{i,2}$. In this model, the necessary condition for the lattice reduction to work (see (20)) becomes:

$$\sum_{i \in I} (\delta_i - c_1) \geq \ell \quad \text{where } \delta_i = \delta_{i,1} + \delta_{i,2}, \tag{37}$$

since $\lambda = 0$ and $N_i = 1$ (there is a single unknown block per nonce), implying:

$$\delta_i \geq \left\lceil \frac{\ell}{\Delta_{max}} \right\rceil + c_1 \tag{38}$$

as a sound condition on each signature. As seen in the previous sections, the CVP algorithm will only succeed in recovering the right solution if the two blocks of k_i are correctly guessed for each signature. In order to select the blocks and their respective sizes $\delta_{i,1}$, $\delta_{i,2}$, we use an approach similar to the one proposed in Section 5.

Let $B_{i,1}$ and $B_{i,2}$ denote the blocks of $\delta_{i,1}$ lsb and $\delta_{i,2}$ msb of k_i. The guessed blocks are defined as:

$$\hat{B}_{i,j} = \underset{x \in \{0,1\}^{\delta_{i,j}}}{\operatorname{argmax}} \operatorname{Pr}(B_{i,j} = x) \tag{39}$$

for $j \in \{1, 2\}$. The block probabilities are then defined as

$$p_{i,j} = \operatorname{Pr}(B_{i,j} = \hat{B}_{i,j}) = \max_x \operatorname{Pr}(B_{i,j} = x)$$

for $j \in \{1, 2\}$ and the probability for one signature is $p_i = p_{i,1} \cdot p_{i,2}$. Then, we select $\delta_{i,1}$ and $\delta_{i,2}$ such that they maximize the value $\gamma_i = p_i^{\frac{1}{\delta_i - c_1}}$.

Clearly, γ_i depends on the sizes $\delta_{i,1}$ and $\delta_{i,2}$ of the blocks $B_{i,1}$ and $B_{i,2}$. We shall denote $\gamma_i(\delta_{i,1}, \delta_{i,2})$ to see γ_i as a function of these sizes. We then set $\delta_{i,1}$ and $\delta_{i,2}$ as $(\delta_{i,1}, \delta_{i,2}) = \operatorname{argmax}_{(\delta_1,\delta_2)} \gamma_i(\delta_1, \delta_2)$ so that $\gamma_i = \max_{(\delta_1,\delta_2)} \gamma_i(\delta_1, \delta_2)$. Unlike for the classic blinding (see Remark 1), the number of selected bits δ_i per signature can be small (it just has to be greater than c_1). Therefore, it is more relevant to select the signatures according to the γ_i value in the case of the Euclidean blinding. In order to still allow several selection trials, we suggest a hybrid approach: we first randomly pick half of the available signatures and then select a subset I among them such that the values $(\gamma_i)_{i \in I}$ are the highest, and such that the constraint (37) is well (tightly) satisfied.

Computation of block probabilities. We now explain how to evaluate (39), *i.e.* how to evaluate the probabilities $\operatorname{Pr}(B_{i,1} = x)$ and $\operatorname{Pr}(B_{i,2} = x)$. For the sake of simplicity, we drop the index i, namely we consider a nonce $k = a \cdot r + b$ for which we know some probability score $\operatorname{Pr}(a_j = 1)$, $\operatorname{Pr}(r_j = 1)$ and $\operatorname{Pr}(b_j = 1)$ for the bits of a, r, and b. Moreover, for some integer v, we shall denote $v_{[j_0:j_1]} = \lfloor \frac{v}{2^{j_0}} \rfloor \bmod 2^{j_1 - j_0}$, *i.e.* the integer value composed of the j_0-th bit to the $(j_1 - 1)$-th bit of v. It is easy to check that the block $B_1 = k_{[0:\delta_1]}$ only depends on the δ_1 lsb of a, r and b. We can then compute the probability $\operatorname{Pr}(B_1 = x)$ as:

$$\operatorname{Pr}(B_1 = x) = \sum_{x,y,z} \chi_1(x, w, y, z) \prod_{j=0}^{\delta_1 - 1} \operatorname{Pr}(a_j = w_j) \operatorname{Pr}(r_j = y_j) \operatorname{Pr}(b_j = z_j) \tag{40}$$

with $\chi_1(x, w, y, z) = 1$ if $x = (w \cdot y + z)_{[0:\delta_1]}$ and $\chi(x, w, y, z) = 0$ otherwise.

For the case of the block $B_2 = k_{[\ell-\delta_2:\ell]}$, we shall denote by a_h and r_h the δ msb of a and r respectively for some δ (*i.e.* $a_h = a_{[\ell-\lambda-\delta:\ell-\lambda]}$ and $r_h = r_{[\lambda-\delta:\lambda]}$). We then have

$$k_{[\ell-2\delta:\ell]} = a_h r_h + c \quad \text{where} \quad c = \left\lfloor \frac{a \cdot r + b - a_h r_h 2^{\ell-2\delta}}{2^{\ell-2\delta}} \right\rfloor . \tag{41}$$

It can be checked that the carry c is lower than $2^{\delta+1}$. Then if we take δ sufficiently greater than δ_2, we get $\tilde{B}_2 = B_2$ with high probability, where $\tilde{B}_2$ denotes the δ_2 msb of $a_h r_h$ (*i.e.* $\tilde{B}_2 = (a_h r_h)_{[2\delta-\delta_2:2\delta]}$). We then approximate

$$\Pr(B_2 = x) \approx \Pr(\tilde{B}_2 = x) = \sum_{x_0=0}^{2^{2\delta-\delta_1}} \Pr(a_h r_h = x_0 + 2^{2\delta-\delta_1} x) \ , \tag{42}$$

where the probability mass function of $a_h r_h$ satisfies

$$\Pr(a_h r_h = x) = \sum_{w,y} \chi_2(x, w, y) \prod_{j=0}^{\delta-1} \Pr(a_{\ell-\lambda-\delta+j} = w_j) \Pr(r_{\lambda-\delta+j} = y_j) \tag{43}$$

where $\chi_2(x, w, y) = 1$ if $x = w \cdot y$ and $\chi_2(x, w, y) = 0$ otherwise.

The above approximation is sound as long as we take δ sufficiently greater than δ_2. Since we have $c \leq 2^{\delta+1}$, it can be checked that the approximation error is upper bounded by $2^{\delta_1+1-\delta}$. We shall then take δ such that the latter approximation error does not impact the selection of the maximal probability $\Pr(B_2 = x)$.

6 Experimental Results

This section provides experimental results: we have implemented our attacks against ECDSA on the ANSSI 256-bit elliptic curve (*i.e.* $\ell = 256$) with the two considered blinding schemes for 3 different random sizes, specifically $\lambda \in \{16, 32, 64\}$. We considered a three-dimensional leakage model with unbalanced multivariate SNR $\theta = \alpha \cdot (0.5, 1, 2)$, where $\alpha \in \{1.5, 2.0\}$. For each setting, the used signatures and blinded nonces were randomly generated, and the corresponding likelihood scores were simulated from the multivariate SNR θ as shown in Proposition 1. We then applied our filtering methods to select the best blocks of bits from which we constructed the Howgrave-Graham and Smart lattice with the constant c_1 set to 4, and we checked whether the correct solution was well retrieved by the CVP algorithm. We experimented our attacks with n_{sig} available signatures and n_{tr} trials for the subset selection (and the underlying lattice reduction) for $(n_{sig}, n_{tr}) \in \{(10, 1), (20, 5), (20, 10), (100, 10), (100, 50), (100, 100)\}$.

The obtained success rates are reported in Table 2. The lattice reduction worked almost every time the selected bits were correctly guessed (we noticed only a few lattice failures in all the performed experiments). These results suggest

Table 2. Success rate of our attacks.

(n_{sig}, n_{tr})		(10,1)	(20, 5)	(20, 10)	(100, 10)	(100, 50)	(100, 100)
Classic blinding							
	$\lambda = 16$	13.5 %	38.3 %	54.0 %	70.1 %	99.0 %	99.9 %
$\alpha = 1.5$	$\lambda = 32$	3.5 %	13.6 %	22.7 %	27.8 %	73.9 %	91.9 %
	$\lambda = 64$	0.2 %	0.6 %	1.2 %	1.5 %	6.2 %	11.7 %
	$\lambda = 16$	91.2 %	99.9 %				
$\alpha = 2$	$\lambda = 32$	90.5 %	99.5 %	100 %	100 %	100 %	100 %
	$\lambda = 64$	85.7 %	99.3 %				
Euclidean blinding							
	$\lambda = 16$						
$\alpha = 1.5$	$\lambda = 32$	0 %	0 %	0 %	0 %	0 %	0 %
	$\lambda = 64$						
	$\lambda = 16$	0.7 %	3.1 %	5.8 %	42.8 %	76.8 %	83.3 %
$\alpha = 2$	$\lambda = 32$	0.1 %	0.4 %	0.8 %	41.1 %	74.9 %	82.6 %
	$\lambda = 64$	0.1 %	0.4 %	1.0 %	40.2 %	75.0 %	82.8 %

that the classic blinding is more sensitive to our attack than the Euclidean blinding. However, one should note that the classic blinding is inefficient when the group order is sparse. We also observe that for the Euclidean blinding, the random size λ has a small impact on the resulting success rate, which is not surprising since it can be checked from Section 5 that this parameter is not expected to play a significant role.

Acknowledgments. The authors were supported in part by the French ANR JCJC ROMAnTIC project (ANR-12-JS02-0004).

References

[Bab86] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[Ble00] D. Bleichenbacher. On the generation of one-time keys in dl signature schemes. Presentation at IEEE P1363 Working Group meeting, November 2000. Unpublished.

[BV15] A. Bauer and D. Vergnaud. Practical key recovery for discrete-logarithm based authentication schemes from random nonce bits. In *CHES 2015*, *LNCS* 9293, pages 287–306. Springer, Heidelberg, September 2015.

[BvdPSY14] N. Benger, J. van de Pol, N. P. Smart, and Y. Yarom. "ooh aah... just a little bit": A small amount of side channel can go a long way. In *CHES 2014*, *LNCS* 8731, pages 75–92. Springer, Heidelberg, September 2014.

[CJ03] M. Ciet and M. Joye. (Virtually) free randomization techniques for elliptic curve cryptography. In *ICICS 03*, *LNCS* 2836, pages 348–359. Springer, Heidelberg, October 2003.

[Cop96a] D. Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *EUROCRYPT'96*, *LNCS* 1070, pages 178–189. Springer, Heidelberg, May 1996.

[Cop96b] D. Coppersmith. Finding a small root of a univariate modular equation. In *EUROCRYPT'96*, *LNCS* 1070, pages 155–165. Springer, Heidelberg, May 1996.

[Cor99] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *CHES'99*, *LNCS* 1717, pages 292–302. Springer, Heidelberg, August 1999.

[CRR03] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *CHES 2002*, *LNCS* 2523, pages 13–28. Springer, Heidelberg, August 2003.

[ElG84] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84*, *LNCS* 196, pages 10–18. Springer, Heidelberg, August 1984.

[FGR13] J.-C. Faugère, C. Goyet, and G. Renault. Attacking (EC)DSA given only an implicit hint. In *SAC 2012*, *LNCS* 7707, pages 252–274. Springer, Heidelberg, August 2013.

[GGH97] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *CRYPTO'97*, *LNCS* 1294, pages 112–131. Springer, Heidelberg, August 1997.

[GJM$^+$11] R. R. Goundar, M. Joye, A. Miyaji, M. Rivain, and A. Venelli. Scalar multiplication on Weierstraß elliptic curves from Co-Z arithmetic. *J. Cryptographic Engineering*, 1(2):161–176, 2011.

[HM09] C. Herbst and M. Medwed. Using templates to attack masked montgomery ladder implementations of modular exponentiation. In *WISA 08*, *LNCS* 5379, pages 1–13. Springer, Heidelberg, September 2009.

[HMHW09] M. Hutter, M. Medwed, D. Hein, and J. Wolkerstorfer. Attacking ECDSA-enabled RFID devices. In *ACNS 09*, *LNCS* 5536, pages 519–534. Springer, Heidelberg, June 2009.

[HS01] N. Howgrave-Graham and N. P. Smart. Lattice attacks on digital signature schemes. *Des. Codes Cryptography*, 23(3):283–290, 2001.

[IMT02] T. Izu, B. Möller, and T. Takagi. Improved elliptic curve multiplication methods resistant against side channel attacks. In *INDOCRYPT 2002*, *LNCS* 2551, pages 296–313. Springer, Heidelberg, December 2002.

[JY03] M. Joye and S.-M. Yen. The Montgomery powering ladder. In *CHES 2002*, *LNCS* 2523, pages 291–302. Springer, Heidelberg, August 2003.

[KJJ99] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO'99*, *LNCS* 1666, pages 388–397. Springer, Heidelberg, August 1999.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(177):203–209, 1987.

[Koc96] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO'96*, *LNCS* 1109, pages 104–113. Springer, Heidelberg, August 1996.

[LLL82] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[LPS04] P. J. Leadbitter, D. Page, and N. P. Smart. Attacking DSA under a repeated bits assumption. In *CHES 2004*, *LNCS* 3156, pages 428–440. Springer, Heidelberg, August 2004.

[MHMP13] E. D. Mulder, M. Hutter, M. E. Marson, and P. Pearson. Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA. In *CHES 2013*, *LNCS* 8086, pages 435–452. Springer, Heidelberg, August 2013.

[Mil86] V. S. Miller. Use of elliptic curves in cryptography. In *CRYPTO'85*, *LNCS* 218, pages 417–426. Springer, Heidelberg, August 1986.

[MO09] M. Medwed and E. Oswald. Template attacks on ECDSA. In *WISA 08*, *LNCS* 5379, pages 14–27. Springer, Heidelberg, September 2009.

[Mon87] P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Math. Comput.*, 48:243–264, 1987.

[Nat00] National Institute of Standards and Technology. *FIPS PUB 186-2: Digital Signature Standard (DSS)*. National Institute for Standards and Technology, Gaithersburg, MD, USA, January 2000.

[NS02] P. Q. Nguyen and I. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15(3):151–176, 2002.

[NS03] P. Q. Nguyen and I. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Des. Codes Cryptography*, 30(2):201–217, 2003.

[Sch91] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[TJ15] M. Tunstall and M. Joye. The distributions of individual bits in the output of multiplicative operations. *Cryptography and Communications*, 7(1):71–90, 2015.

[WW07] I. Wegener and P. Woelfel. New results on the complexity of the middle bit of multiplication. *Computational Complexity*, 16(3):298–323, 2007.

A Using Exhaustive Search

In order to increase the success probability of our attack, we can make use of exhaustive search to guess certain bits in a signature. Specifically, we can select m bits among all the signatures for which we shall try the 2^m possible values and apply the previously defined lattice attack. To select the guessed bits in a signature we follow the same approach as previously but with a new dimension for the parameters: the number of bits allowed to be exhaustively guessed. Specifically, we shall compute the optimal parameters γ_i, J_i and N_i for each possible $m_i \leq m_{max}$, where m_i is the number of bits exhaustively guessed in the i-th signature and m_{max} is the maximal value for m. We hence get a new constraint to the selection of the signatures, that is:

$$\sum_{i=0}^{n} m_i \leq m_{max} \ . \tag{44}$$

Let us introduce the function π_i defined for every $J \subseteq [\![0, \ell + \lambda - 1]\!]$ and $m \in \mathbb{N}$ as:

$$\pi_i(J, m) = \max_{\substack{J' \subseteq J \\ |J'| = |J| - m}} \prod_{j \in J'} p_{i,j} \ . \tag{45}$$

Namely, $\pi_i(J, m)$ is the maximal product of $|J| - m$ probabilities among $(p_{i,j})_{j \in J}$. This gives the probability of correctly guessing the bits $(a_{i,j})_{j \in J}$ while m among them are exhaustively guessed. We then define

$$J_i(m, N) = \operatorname*{argmax}_{\substack{J; |J| = f(N) \\ g(J) \leq N}} \pi_i(J, m) \tag{46}$$

and

$$p_i(m,N) = \pi_i(J_i(m,N),m)\ ,\quad \gamma_i(m,N) = p_i(m,N)^{\frac{1}{f(N)-\lambda-c_1 N}}\ , \tag{47}$$

from which we set

$$\gamma_i(m) = \max_N \gamma_i(m,N)\ ,\quad N_i(m) = \operatorname*{argmax}_N \gamma_i(m,N)\ ,\ \text{and}\ J_i(m) = J_i(m,N_i(m))\ . \tag{48}$$

Eventually, we select I and $(J_i(m_i))_{i\in I}$ such that (44) and

$$\ell \le \sum_{i\in I} |J_i(m_i)| - \lambda - c_1 N_i \tag{49}$$

are both satisfied.

Evaluating Equation (46)**.** We now explain how to evaluate the function

$$(m,N) \mapsto \max_{\substack{J;|J|=f(N)\\ g(J)\le N}} \pi_i(J,m)\ , \tag{50}$$

from which getting the argmax is then straightforward. For such a purpose, we extend the maxp function of Section 4 to deal with the m parameter *i.e.* the number of bits that can be exhaustively guessed. Here, maxp maps a quadruple (j,δ,N,m) to the above max with the condition $J \subseteq [\![j,\ell+\lambda-1]\!]$, $|J|=\delta$, and $g(J)\le N$, so that the target function in (50) can be rewritten as:

$$(m,N) \mapsto \mathsf{maxp}(0,f(N),N,m) \tag{51}$$

The difference with the original method is that in a recursive call to the maxp function, one can spend v out of m bits to be exhaustively guessed. Specifically, the recurrence relation becomes:

$$\mathsf{maxp}(j,\delta,N,m) = \max(\mathsf{block}(j,\delta,N,m),\mathsf{next}(j,\delta,N,m)) \tag{52}$$

with

$$\begin{aligned}\mathsf{block}(j,\delta,N,m) &= \max_{\substack{w\in[\![1,\delta]\!]\\ v\in[\![0,m]\!]}} \pi_i([\![j,j+w-1]\!],v)\\ &\qquad\qquad \times \mathsf{maxp}(j+w+1,\delta-w,N-1,m-v)\\ \mathsf{next}(j,\delta,N,m) &= \mathsf{maxp}(j+1,\delta,N,m)\end{aligned}$$

The recursion tail is pretty similar to the original case:

$$\mathsf{maxp}(j,\delta,N,m) = \begin{cases} 0 & \text{if } j+\delta > \ell+\lambda\\ \pi_i([\![j,\ell+\lambda]\!],m) & \text{if } j+\delta = \ell+\lambda-1\\ 0 & \text{if } N=0 \text{ and } \delta>0\\ 1 & \text{if } \delta = 0\end{cases} \tag{53}$$

Only the second case changes, where the m left bits of exhaustive search are spend for the final block.