Side-channel Attacks on Blinded Scalar Multiplications Revisited

Thomas Roche[1], Laurent Imbert[2], and Victor Lomné[1]

[1] NinjaLab, Montpellier, France (`https://ninjalab.io`)
[2] LIRMM, CNRS, University of Montpellier, Montpellier, France

Abstract. In a series of recent articles (from 2011 to 2017), Schindler *et al.* show that exponent/scalar blinding is not as effective a countermeasure as expected against side-channel attacks targeting RSA modular exponentiation and ECC scalar multiplication. Precisely, these works demonstrate that if an attacker is able to retrieve many randomizations of the same secret, this secret can be fully recovered even when a significative proportion of the blinded secret bits are erroneous. With a focus on ECC, this paper improves the best results of Schindler *et al.* in both the generic case of random-order elliptic curves and the specific case of structured-order elliptic curves. Our results show that larger blinding material and higher error rates can be successfully handled by an attacker in practice. This study also opens new directions in this line of work by the proposal of a three-steps attack process that isolates the attack critical path (in terms of complexity and success rate) and hence eases the development of future solutions.

1 Introduction

Nowadays, all modern tamper-resistant implementations of public-key algorithms embed relatively cheap, yet very strong countermeasures based on various randomization strategies. As a consequence, single-trace horizontal attacks have gained more and more attention from the side-channel community.

Single trace horizontal attacks apply to both elliptic curve scalar multiplication and modular exponentiation (RSA). Implemented in a supervised or non-supervised setup, they provide the attacker with a randomized, or blinded scalar (resp. exponent) from the observation of a single scalar multiplication or exponentiation. Although these attacks do not yield the original scalar (resp. exponent), the disclosure of a blinded value may allow an attacker to counterfeit digital signatures or impersonate any party in a key exchange protocol.

This ultimate attack thus renders scalar (resp. exponent) randomization useless. However, it requires a very high signal-to-noise ratio to be successful in practice. Many recent publications claim successful single trace horizontal attacks on secure RSA or ECC [2, 8–12, 17, 19]. These attacks do not usually recover the whole blinded value. The missing bits are eventually recovered using brute-force. Therefore, the number of incorrect bits must remain relatively small for the attack to be successful. In single-trace horizontal attacks, this number of incorrect bits is dictated by the so-called bit error rate of the attack.

In this work, we consider the case where brute-forcing the incorrect bits is impracticable. We focus on ECC scalar multiplication but our analysis for random-order elliptic curves easily extends to RSA exponentiation. We assume that the attacker can observe several scalar multiplications with the same long-term secret scalar but each execution uses fresh randoms for scalar blinding. A typical example of such a context occurs in the public key generation of ECC cryptosystems. The attacker requests from a device many generations of the public key corresponding to the private key securely stored inside the device. We will also assume that the scalar randomization is done following [4] by adding to the secret scalar a random multiple of the elliptic curve order.

The first paper in the literature to tackle this problem is the seminal work of Schindler and Itoh [13] which exhibits a very efficient attack (in terms of number of traces and computational effort) when small blinding factors r are used. Over the past five years, this result was improved [14], applied to specific elliptic curves [5,15] and to RSA with CRT [16]. In the present paper, we expand this line of results by suggesting several improvements that make it possible to recover scalars blinded with large random factors (> 32 bits), and high bit error rates ($> 10\%$).

1.1 Preliminaries and Notations

In the following, we consider an elliptic curve defined over the finite field $\mathbb{F}_p$, with p a K-bit prime (typically $K = 256$). E denotes the order of the curve and d is the secret scalar, target of the attack. Both E and d can be represented on K bits. The term msb (resp. lsb) will be used to shorten *most* (resp. *least*) *significant bits.*

For each scalar multiplication, the scalar d is blinded by adding a random multiple of the group order, i.e. $d_\ell = d + r_\ell \times E$, where r_ℓ is an R-bit random value. The blinded scalar d_ℓ is then represented on $K + R$ bits.

The attaker observes N scalar multiplications. These N side-channel observations, called *traces*, are denoted $\{T_\ell\}_{\ell<N}$[3].

For each trace T_ℓ, the attacker's horizontal side-channel attack outputs a *noisy* blinded scalar, denoted $\tilde{d_\ell}$. For all bit index $i < K + R$, it is assumed that the probability ϵ_b for bit $\tilde{d_\ell}[i]$ to be erroneous, called *bit error rate*, is independent of both ℓ and i. Depending on the context (supervised or non-supervised horizontal attacks) ϵ_b is considered known or unknown to the attacker.

In [13], the authors introduce a cornerstone tool for all subsequent attacks called *Non-Adjacent-Form distinguisher*. Given two noisy blinded scalars $(\tilde{d_i}, \tilde{d_j})$ and two guesses $(\hat{r}_i, \hat{r}_j)$ on their respective blinding factors, the distinguisher provides a very robust way to assert the correctness of the guesses. We define

[3] A more formal notation would be $\{T_\ell\}_{\ell\in\mathbb{Z};0\leqslant\ell<N}$.

$D_{i,j}^{\mathrm{NAF}}$ as follows:

$$D_{i,j}^{\mathrm{NAF}} = \mathrm{HW}(\mathrm{NAF}(\tilde{d}_i - \hat{r}_i \times E - (\tilde{d}_j - \hat{r}_j \times E))),$$

where NAF() is the non-adjacent form encoding (see *e.g.* [7]) and HW() is the natural extension of the Hamming Weight function to signed-digit representations.

Schindler and Itoh observed[4] that $D_{i,j}^{\mathrm{NAF}}$ is significantly smaller for correct values $(\hat{r}_i, \hat{r}_j)$. A detailed study of the distinguisher behaviour with respect to $K+R$ and ϵ_b can be found in [13, 14], it allows to choose a bound $\mathcal{B}_{\epsilon_b,K+R}$ such that $D_{i,j}^{\mathrm{NAF}} < \mathcal{B}_{\epsilon_b,K+R}$ implies that $(\hat{r}_i, \hat{r}_j)$ are correct with high probability. Clearly, this distinguisher will be more effective as K and R increase and ϵ_b decreases. The non-adjacent-form distinguisher proves to be sufficient for parameters considered in the present study ($K = 256$, $32 \leqslant R \leqslant 128$, $0.1 \leqslant \epsilon_b \leqslant 0.25$).

1.2 Overall Attack Process

Our attack context is the gathering of three independent steps. Our contributions are solely related to the second step and, for completeness, we briefly describe the whole attack process below.

Step 1: The attacker acquires N traces corresponding to N independent scalar multiplications and performs a horizontal attack for each of them. The output of this first step is a set of noisy blinded scalars $\{\tilde{d}_\ell\}_{\ell<N}$ together with a bit error rate ϵ_b. In the supervised setting[5] (see *e.g.* [1, 2, 19]) the attacker possesses a good estimation of the bit error rate ϵ_b. Given ϵ_b the attacker knows beforehand the number of acquisitions N that must be performed to have good chance of success. In the more general unsupervised setting (see *e.g.* [8–12, 17]), the access to a training device is not possible. The attacker acquires as many traces as possible and induces a maximal value for ϵ_b that can be handled through the attack. In both cases, this first step provides the attacker with N noisy blinded scalars together with a gross value for ϵ_b.

Step 2: From each noisy blinded scalar $\tilde{d}_\ell$, the attacker guesses the blinding factor r_ℓ or discards the corresponding data from the attack process. The output of this filtering step is a subset $\{\tilde{d}_\ell\}_{\ell\in J}$ along with guessed blinding factors $\{r_\ell\}_{\ell\in J}$ for some $J \subset (\mathbb{Z} \cap [0, N-1])$. All r_ℓ do not have to be correct but some of them must be correctly guessed.

[4] More precisely, they use this distinguisher to find blinding factor collisions ($r_i = r_j$) by computing $\mathrm{HW}(\mathrm{NAF}(\tilde{d}_i - \tilde{d}_j))$ (which corresponds to $D_{i,j}^{\mathrm{NAF}}$ when $r_i = r_j$).

[5] A learning phase is conducted prior to the attack on a similar device where scalar multiplication inputs and randoms can be chosen, *e.g.* a template building or a deep-learning training phase.

Step 3: The last step of the attack recovers the secret scalar d from $\{\tilde{d}_\ell\}_{\ell \in J}$ and $\{r_\ell\}_{\ell \in J}$. These inputs may first be trimmed off using *e.g.* the NAF distinguisher. (The pair $(\tilde{d}_i, r_i)$ is removed if, for all $j \neq i$, $D^{\mathrm{NAF}}_{i,j}$ is larger than the predefined bound $\mathcal{B}_{\epsilon_b, K+R}$.) After this optional pruning step, a *vertical* side-channel attack can be mounted on the remaining traces. Such an attack is described in [5] for specific elliptic curves but can be extended straightforwardly in our context to any elliptic curve and RSA. In Appendix A, we propose a simple vertical attack to confirm this claim and show that few tens of correctly guessed blinding factors are enough to successfully recover the secret scalar (resp. exponent).

1.3 Paper Organization and Contributions

This work focuses on improvements in *Step 2* when $R > 32$. The first contribution applies to general elliptic curves without hypothesis on the curve order form. It is described in Section 2. This new proposal outperforms the current best attacks by Schindler and Wiemers [14,16], *e.g.* we show that configurations where $(R = 64, \epsilon_b = 0.2)$ and $(R = 96, \epsilon_b = 0.15)$ can be successfuly attacked in practice. The second contribution applies to the specific case of elliptic curves whose order is close to a power of 2. Our strategy and results are presented in Section 3. Compared to the best known attack [15], our simulations show significant increases in the success rates for blinding factors up to $K/2$.

2 On Random-Order Elliptic Curves

In this section we consider elliptic curves without assumption on the form of their order. Based on previous work, our results can be extended to RSA implementations without CRT[6] and even implementations with CRT[7].

We will first briefly describe the *Alternate Attack* [14] (also described in [16]). To the best of our knowledge, this attack is the only known attack in the literature that works in practice for non-negligible values of ϵ_b and large blinding factors ($R > 32$, typically $R = 64$ or $R = 96$). In Section 2.2, we present a new attack which, compared to [14], offers two advantages: *1. its simplicity.* It allows to exhibit the computational cost of the algorithm for arbitrary chosen success rate without having to go through costly simulations. Hence providing a better understanding on the capability of the attack given the attacker's computational effort. *2. its efficiency.* For an equivalent computational cost, it handles larger values of R and/or larger values of ϵ_b than [14]. This comes at a price: increasing the number of observed scalar multiplications (*i.e.* N).

2.1 Previous Work

Let us briefly recall the Alternate attack from Schindler and Wiemers (for more details see [14, Algorithm 3], also called *sub-attack I*).

[6] relying on the fact that an RSA modulus n and $\phi(n)$ share half of their msb. See [14] for more details.
[7] the authors of [16] managed to recover $2R$ msb of $\phi(p)$ based on continued fractions.

Given a blinded scalar d_ℓ, Schindler and Wiemers define $\alpha_\ell = \lfloor d_\ell / 2^{K-1} \rfloor$ (thus $\alpha_\ell < 2^{R+1}$) and $\beta_\ell = d_\ell \bmod 2^w$ (for a chosen window size $w < K$). Since $d_\ell = d + r_\ell \times E$, they demonstrate that the w lsb of d verify the following equation:

$$d \bmod 2^w = \left(\alpha_\ell 2^{K-1} \bmod E + \beta_\ell - \omega_\ell E\right) \bmod 2^w,$$

where ω_ℓ takes an unknown value in $\{0,1\}$.

From noisy blinded scalars $\tilde{d}_\ell$, we have access to noisy versions of α_ℓ (say $\tilde{\alpha}_\ell$) and β_ℓ (say $\tilde{\beta}_\ell$). The idea is then to brute-force all error patterns (of Hamming Weight t) over $\tilde{\alpha}_\ell$ and $\tilde{\beta}_\ell$ (t must be carefully chosen given ϵ_b) as well as the values of ω_ℓ. Each hypothesis on $(\alpha_\ell, \beta_\ell, \omega_\ell)$ will provide a candidate x for $d \bmod 2^w$. Over all N noisy blinded scalars, the correct $d \bmod 2^w$ should be in the best ranked candidates x (those that appeared the most during the whole brute-force).

If successful this attack outputs the w lsb of d, for w typically as large as R. To finish the attack (and recover the remaining bits of d), the authors of [14] apply iteratively another attack (so-called *sub-attack II*). In our setup (see Section 1.2 for a description of the overall attack process), we would more likely use the knowledge of $d \bmod 2^w$ to rewind the Alternate Attack and find the most likely blinding factors r_ℓ. Indeed, for some scalars, the correct value of $(\alpha_\ell, \beta_\ell, \omega_\ell)$ must have been found during the brute-force and can be collected from the knowledge of $d \bmod 2^w$. Obtaining the random factor r_ℓ is then straightforward as $r_\ell = \lfloor \alpha_\ell 2^{K-1} / E \rfloor + \omega_\ell$. Then one would apply Step 3 of the overall attack process by suppressing the erroneous values of r_ℓ based on the *NAF distinguisher* and following the approach described in Annex A.

The computational cost of the *sub-attack I* algorithm is $2M_0N$ predictions of triplets $(\alpha_\ell, \beta_\ell, \omega_\ell)$, where $M_0 = \binom{R+w}{t}$ with t the number of erroneous bits tested by the brute-force over the $\sim R+w$ bits of $(\alpha_\ell, \beta_\ell)$. The authors estimate the correct values of t, w and N for successful attacks and provide the following results (copied from [14, Table 12]) that gives the maximum value of R for various attack configurations.

N	$2M_0N$	$\epsilon_b = 0.10$	$\epsilon_b = 0.15$	$\epsilon_b = 0.20$
10	$\leqslant 2^{40}$	36	16	4
1000	$\leqslant 2^{45}$	96	52	28
100000	$\leqslant 2^{50}$	136	76	44

Table 1. Maximum R-values for Schindler and Wiemers's Algorithm 3 [14, Table 12]

2.2 A New Generic Algorithm

From the bit error rate ϵ_b and R it is easy to choose N such that there is a good chance that at least 2 noisy blinded scalars are correct on their $R+1$ msb (*i.e.* $\tilde{\alpha}_\ell = \alpha_\ell$). Indeed, in the idealized hypothesis where each bit of each $\tilde{d}_\ell$ is erroneous with probability ϵ_b independently from all other bits, the number of erroneous bits in the $R+1$ msb of each $\tilde{d}_\ell$ follows a Binomial distribution with parameters $(R+1, \epsilon_b)$. Therefore, on average, the number of observations needed to acquire n noisy blinded scalars with t erroneous bits or less in their $R+1$ msb is given by $N = n / \mathrm{CDF}_{(R+1,\epsilon_b)}(t)$, where $\mathrm{CDF}_{(R+1,\epsilon_b)}(t)$ is the cumulative distribution function of the Binomial distribution with parameters $(R+1, \epsilon_b)$. Equivalently, if the attacker observes N noisy blinded scalars, the average number of observations that have t or less erroneous bits in their $R+1$ msb is given by $n = N \times \mathrm{CDF}_{(R+1,\epsilon_b)}(t)$.

Now, let us assume that N is large enough with respect to $(R+1, \epsilon_b)$ such that there exists (with good probability) two indexes $i < j < N$ such that the $R+1$ most significant bits of $\tilde{d}_i$ and $\tilde{d}_j$ are all correct. The idea is to identify the pair (i, j) based on the *NAF distinguisher* (see Section 1.1): for all $l < N$, estimate the blinding factor of $\tilde{d}_l$ as follows

$$\tilde{r}_l^{\omega_l} = \left\lfloor \tilde{\alpha}_l 2^{K-1} / E \right\rfloor + \omega_l,$$

where $\tilde{\alpha}_l$ is the $R+1$ msb of $\tilde{d}_l$ and ω_l takes value 0 or 1. The attacker possesses then two blinding factor candidates for each $\tilde{d}_l$. Also, by hypothesis, there exists (ω_i, ω_j) such that $r_i = \tilde{r}_i^{\omega_i}$ and $r_j = \tilde{r}_j^{\omega_j}$. Computing the *NAF distinguisher* on all pairs of noisy blinded scalars for both of their blinding factors, one can identify the pair (i, j) if they do exist ($D_{i,j}^{\mathrm{NAF}}$ is significantly small compared to $D_{k,l}^{\mathrm{NAF}}$ where $r_k \neq \tilde{r}_k^{\omega_k}$ or $r_l \neq \tilde{r}_l^{\omega_l}$). This costs $4\frac{N(N-1)}{2}$ computations of D^{NAF}.

Once the pair (i, j) is found, the corresponding masks r_i and r_j are known. It is now possible to run through the remaining $N-2$ noisy blinded scalar, for each of them, one can brute-force all error patterns with t erroneous bits or less among the $R+1$ msb. For each error pattern, two blinding factors $\{\tilde{r}_l^{\omega_l}\}_{\omega_l \in \{0,1\}}$ are computed from $\tilde{d}_l$ and the corresponding $D_{i,l}^{\mathrm{NAF}}$ (or $D_{j,l}^{\mathrm{NAF}}$) can be estimated asserting if the correct blinding factor r_l was found. t must be chosen such that, at the end of this step, enough correct blinding factors are found (such that, *e.g.* the procedure described in Appendix A can be applied successfully). This step costs $2(N-2) \times \binom{R}{t}$ computations of D^{NAF}. Since N was chosen large enough to yield the initial error-free scalars (i and j), small values of t (typically 1 or 2) are enough in practice to find few tens of correct r_l. The overall complexity of the algorithm is then dominated by $2N(N-1)$ D^{NAF} computations in practice.

Parameter Choices and Comparison with Alternate Attack [14] Table 2 displays the number of noisy blinded scalars (N) such that, on average, at least 2 such scalars are error-free on their $R+1$ msb[8]. Let us emphasize that these

[8] N is estimated as follows: $N = 2 / \mathrm{CDF}_{(R+1,\epsilon_b)}(0)$. To estimate the attack success rate, *i.e.* the chance that actually 2 or more noisy blinded scalars are correct on their $R+1$ msb, it is equal to $s = 1 - \mathrm{CDF}_{(N, \mathrm{CDF}_{(R+1,\epsilon_b)}(0))}(1)$. Our values of N imply then a 60% success rate. Taking $2N$ instead of N in Table 2 would increase the attack success rate to 90%.

estimations are independent from both the size K of E and d. However the attack itself depends on K since the NAF distinguisher will work better as K increases. For instance, with $K = 256$, and bit error rates $\epsilon_b > 0.25$ the distinguisher looses effectivness and becomes totally ineffective when reaching $\epsilon_b = 0.4$. These off-limits cases are identified in Table 2 with blue cells. Also, one needs to take into account the attack complexity which grows quadratically with N. Gray cells in Table 2 identify parameters that would lead to 2^{60} or larger number of D^{NAF} computations, making this attack hardly possible in practice.

R \ ϵ_b	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
32	$2^{3.5}$	$2^{6.0}$	$2^{8.7}$	$2^{11.6}$	$2^{14.7}$	$2^{18.0}$	$2^{21.5}$	$2^{25.3}$
64	$2^{5.8}$	$2^{10.9}$	$2^{16.2}$	$2^{21.9}$	$2^{28.0}$	$2^{34.4}$	$2^{41.4}$	$2^{48.9}$
96	$2^{8.2}$	$2^{15.7}$	$2^{23.7}$	$2^{32.2}$	$2^{41.3}$	$2^{50.9}$	$2^{61.3}$	$2^{72.5}$
128	$2^{10.5}$	$2^{20.6}$	$2^{31.2}$	$2^{42.5}$	$2^{54.5}$	$2^{67.4}$	$2^{81.2}$	$2^{96.1}$

Table 2. Number of scalar multiplications (for a success rate > 50%)

Without an efficient way to estimate the success rate of the Alternate Attack from [14], it is difficult to thoroughtly compare the two attacks. Nevertheless, one can observe from Table 1 that $(R > 44, \epsilon_b = 0.2)$ and $(R > 76, \epsilon_b = 0.15)$ cannot be handled successfully by the Alternate Attack whereas we have shown that $(R = 64, \epsilon_b = 0.2)$ and $(R = 96, \epsilon_b = 0.15)$ are achievable in practice when several millions of scalar multiplications can be observed.

3 On Structured-Order Elliptic Curves

In this section, we focus on elliptic curves with structured orders[9].

3.1 Previous Works

In [15], Schindler and Wiemers study elliptic curves with order of the form $E = 2^K \pm E_0$, where E_0 is close to $2^{K/2}$. This case is pretty common in cryptography when the base field is defined using a pseudo-Mersenne prime for efficiency reasons. Most of the EC standards are of this form, *e.g.* SEC2 curves [18], NIST curves [6].

[9] Therefore this study does not apply to RSA

A Divide and Conquer Algorithm Schindler and Wiemers observe that the problem of solving the N noisy blinded scalars can be done using a divide and conquer algorithm. This observation leads to a much more robust decoding algorithm than in the general case. Indeed, a blinded scalar d_ℓ with blinding factor r_ℓ can be written as follows:

$$\begin{aligned} d_\ell &= r_\ell \times E + d \\ &= r_\ell \times (2^K \pm E_0) + d \\ &= r_\ell \times 2^K + (d \pm r_\ell \times E_0) \end{aligned}$$

Hence, if $d \pm r_\ell \times E_0$ is smaller than 2^K, then the R msb of d_ℓ are exactly the R bits of r_ℓ. As a side remark, if $r_\ell \times E_0$ is smaller than d, then the most significant bits of d are not correctly masked (see *e.g.* [3]).

Now, for a given window size w, if $(d \pm r_\ell \times E_0) < 2^K$, then $d_\ell \bmod 2^w$ and $\lfloor d_\ell / 2^K \rfloor \bmod 2^w$ only involve the known w lsb of E and the unknown w lsb of d and r_ℓ. From this observation, Schindler and Wiemers (see [15]) propose an efficient algorithm to recover the secret d that comprises three phases:

- *Phase 1:* find the R lsb of d as well as the most likely values of the blinding factor r_ℓ for each noisy blinded scalar $\tilde{d}_\ell$.
- *Phase 2:* select the values r_ℓ that are the most likely to be correct based on the NAF distinguisher (see Section 1.1).
- *Phase 3:* recover the full secret scalar d.

Phases 2 and 3 correspond exactly to step 3 of our overall attack scheme described in Section 1.2.

Schindler and Wiemers' Phase 1 Algorithm is described in [15, Algorithm 4] along with several empirical improvments discussed in the next sections. The algorithm processes iteratively over a small sliding window of size w (typically w is 8 or 10). Each iteration consists of two main steps recalled in Algorithm 1 and Algorithm 2 respectively.

In Algorithm 1, the call to EvaluateProbability$(\hat{r}_\ell, \hat{d}_\ell, \tilde{d}_\ell, i, w, \epsilon_b)$ computes the probability of observing $\tilde{d}_\ell$ knowing the error rate ϵ_b and the two w-bit words $\hat{r}_\ell$ and $\hat{d}_\ell$ which correspond respectively to the two w-bit words $\left\lfloor \tilde{d}_\ell / 2^{K+i-1} \right\rfloor \bmod 2^w$ and $\left\lfloor \tilde{d}_\ell / 2^{i-1} \right\rfloor \bmod 2^w$. Hence, we have:

$$\text{EvaluateProbability}(\hat{r}_\ell, \hat{d}_\ell, \tilde{d}_\ell, i, w, \epsilon_b) = \epsilon_b^h (1 - \epsilon_b)^{2w-h},$$

where

$$\begin{aligned} h = \texttt{HammingDistance}(\hat{r}_\ell, \left\lfloor \tilde{d}_\ell / 2^{K+i-1} \right\rfloor \bmod 2^w) + \\ \texttt{HammingDistance}(\hat{d}_\ell, \left\lfloor \tilde{d}_\ell / 2^{i-1} \right\rfloor \bmod 2^w) \end{aligned}$$

```
Parameter : Iteration $i$
Parameter : Window size $w$, bit error rate $\epsilon_b$
Input     : $\{\tilde{d}_\ell\}_{\ell<N}$: $N$ noisy scalars
Input     : $\{\tilde{r}_\ell\}_{\ell<N}$: $i-1$ lsb of the recovered blinding factors
Input     : $d \bmod 2^{i-1}$: $i-1$ lsb of the recovered scalar
Output    : $d^*$: best guess for $d \bmod 2^{w+i-1}$
1  $P \leftarrow$ float 1D array of size $2^w$ initialized with zeros;
2  // For each possible value of the next $w$ bits of the secret scalar;
3  for $\hat{d} \leftarrow 0$ to $2^w - 1$ do
4      // Prediction of the $w+i-1$ lsb of the scalar knowing the first $i-1$ bits;
5      $\bar{d} \leftarrow \hat{d} \times 2^{i-1} + d \bmod 2^{i-1}$;
6      // For each noisy blinded scalar;
7      for $\ell \leftarrow 0$ to $N-1$ do
8          // For each possible value of the next $w$ bits of the random $r_\ell$;
9          for $\hat{r}_\ell \leftarrow 0$ to $2^w - 1$ do
10             $\bar{r}_\ell \leftarrow \hat{r}_\ell \times 2^{i-1} + \tilde{r}_\ell$;
11             // Predict $w+i-1$ lsb of $d_\ell$;
12             $\bar{d}_\ell \leftarrow (\bar{r}_\ell \times E + \bar{d}) \bmod 2^{w+i-1}$;
13             // Define $\hat{d}_\ell$, the $w$ msb of $\bar{d}_\ell$;
14             $\hat{d}_\ell \leftarrow \lfloor \bar{d}_\ell / 2^{i-1} \rfloor$;
15             // Compute the probability of observing $\tilde{d}_\ell$, knowing $\hat{d}$ blinded by $\hat{r}_\ell$;
16             $p \leftarrow$ EvaluateProbability$(\hat{r}_\ell, \hat{d}_\ell, \tilde{d}_\ell, i, w, \epsilon_b)$;
17             $P[\hat{d}] \leftarrow P[\hat{d}] + p$;
18 $d^* \leftarrow \operatorname{argmax}(P) \times 2^{i-1} + (d \bmod 2^{i-1})$;
Return : $d^*$
```

Algorithm 1: Phase 1, Step 1 of [15, Algorithm 4]

After R iterations of Algorithms 1 and 2 the output is $d \bmod 2^R$ if everything went correctly. As stated above, this is the most critical phase in Schindler and Wiemers's algorithm. They propose two empirical approaches to improve both its efficiency and effectivness. We will briefly present them in the next section. However, since these improvements are based on hand-picked thresholds by the authors of [15] without clear explanations on how to choose these limits (we are assuming that these thresholds must be adjusted in a case-by-case manner) we will not take them into account in our study. Nevertheless, since the improvements presented here can be applied on the core algorithms, the empirical improvements can always be added above them. We then focus on the low level algorithms and leave for future work the addition and study of these extra improvments.

Empirical Improvements The first improvement is added to Algorithm 2 to increase the effectivness of the attack. Concretely, the authors add an estimation of the correctness of $\bar{r}_\ell$. When this estimation of correctness goes below a certain threshold, the corresponding noisy blinded scalar $\tilde{d}_\ell$ is removed from the process. The question of how to choose the threshold is not discussed in [15] but several values are proposed depending on the bit error rate ϵ_b and the iteration number i.

The second improvement is dedicated to efficiency. The algorithm cost is dominated by Step 1 (Algorithm 1), its complexity being $O(2^{2w}N)$. The authors

```
Parameter  : Iteration $i$
Parameter  : Window size $w$, bit error rate $\epsilon_b$
Input      : $\{\tilde{d}_\ell\}_{\ell<N}$: $N$ noisy scalars
Input      : $\{\tilde{r}_\ell\}_{\ell<N}$: $i-1$ lsb of the recovered blinding factors
Input      : $d^*$: $w+i-1$ lsb of the recovered scalar from Step 1
Output     : $d \bmod 2^i$
Output     : $\{\tilde{r}_\ell\}_{\ell<N}$: $i$ lsb of the recovered blinding factors
1  // For each noisy blinded scalar;
2  for $\ell \leftarrow 0$ to $N-1$ do
3      $P \leftarrow$ float 1D array of size 2 initialized with zeros;
4      // For each possible value of the next $w$ bits of the random $r_\ell$;
5      for $\hat{r}_\ell \leftarrow 0$ to $2^w$ do
6          $\bar{r}_\ell \leftarrow \hat{r}_\ell \times 2^{i-1} + \tilde{r}_\ell$;
7          // Predict $w+i-1$ lsb of $d_\ell$;
8          $\bar{d}_\ell \leftarrow (\bar{r}_\ell \times E + d^*) \bmod 2^{w+i-1}$;
9          // Define $\hat{d}_\ell$, the $w$ msb of $\bar{d}_\ell$;
10         $\hat{d}_\ell \leftarrow \lfloor \bar{d}_\ell / 2^{i-1} \rfloor$;
11         // Compute the probability of observing $\tilde{d}_\ell$, knowing $d^*$ blinded by $\hat{r}_\ell$;
12         $p \leftarrow \text{EvaluateProbability}(\hat{r}_\ell, \hat{d}_\ell, \tilde{d}_\ell, i, w, \epsilon_b)$;
13         $P[\hat{r}_\ell \bmod 2] \leftarrow P[\hat{r}_\ell \bmod 2] + p$;
14     $\tilde{r}_\ell \leftarrow \text{argmax}(P) \times 2^{i-1} + \tilde{r}_\ell$;
15 $d^* \leftarrow d^* \bmod 2^i$;
   Return     : $d^*, \{\tilde{r}_\ell\}_{\ell<N}$
```

Algorithm 2: Phase 1, Step 2 of [15, Algorithm 4]

propose to reduce the number of treated noisy blinded scalars in this step and apply the second step to all noisy blinded scalars. The idea is that, if costly, Step 1 is more robust than Step 2 and therefore does not need all the N noisy blinded scalars to correctly guess $d \bmod 2^{w+i-1}$. The authors propose, again without justification, the "good" number of noisy blinded scalars to be used in Step 1 depending on the bit error rate ϵ_b and the iteration number i.

The above improvements were not tested in this paper. However, one can easily see that they can be applied pretty much similarly to our algorithms with adjusted thresholds.

3.2 Some Results

It is shown in [15] that Algorithms 1 and 2 allow to correct noisy blinded scalars with large values of R, typically $\geqslant 64$ and large error rates $0.1 \leqslant \epsilon_b \leqslant 0.15$. This result is very important since before [15], a value of $R = 64$ was considered perfectly safe from a side-channel point of view.

One crucial parameter of these algorithms is the choice of the window size w since the robustness of the procedure increases with w. However, since the algorithm complexity is dominated by $O(2^{2w}N)$, w cannot be very large either. Figure 1 provides simulation results for various values of w of Algorithms 1 and 2. It gives the average number of bits of d guessed correctly before a wrong bit appears, as a function of the number of traces N. These simulations were done with $K = 256$, $R = 64$ and $\epsilon_b = 0.15$ for curve `secp-256-k1` [18] (aka the Bitcoin's curve).

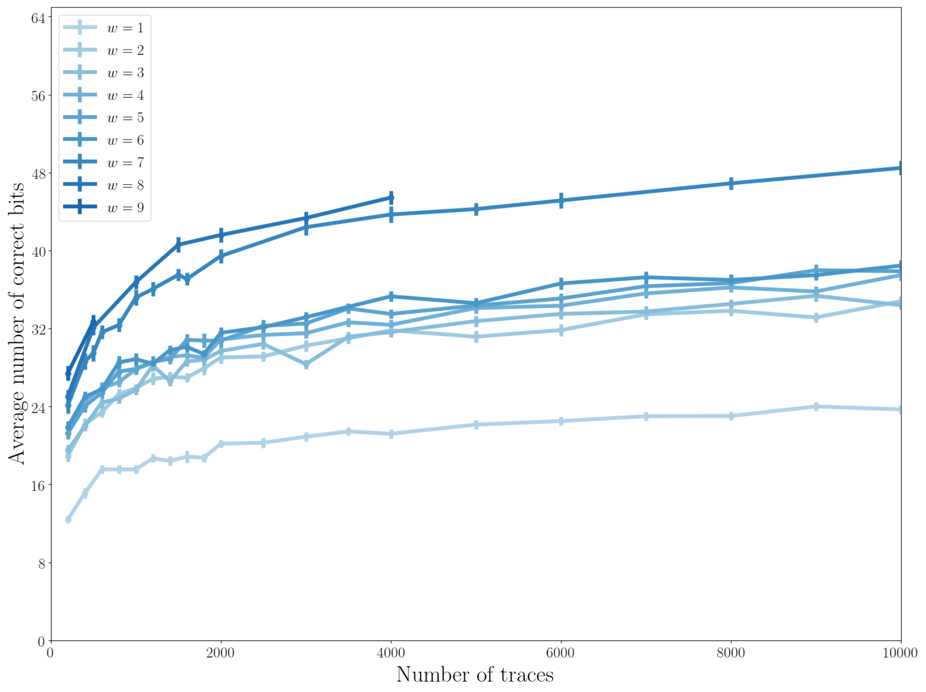

Fig. 1. Simulations for $K = 256, R = 64, \epsilon_b = 0.15$ on curve `secp-256-k1`.

In Figure 1, we represent mean values over 50 executions of the algorithms. Standard deviations to the average results are illustrated by error bars. These simulations are extremely time consuming as w increases. This is why some results are missing for $w > 7$. This is probably why the simulation results in [15] are scattered over a few parameters. We believe that Figure 1 provides a complementary point of view on the efficiency of the correction algorithm of [15][10]. Notably, it is interesting to remark that the impact of the window size is not regular and that window sizes ranging from 3 to 6 produce similar success rates.

We will see in next Section how the algorithms can be improved in both efficiency and effectivness.

3.3 Improved Algorithms

First Observations As remarked earlier (and in [15]) w cannot be too small for the algorithm to work. The reason is that the probability estimation (from the call to EvaluateProbability() in Algorithm 1) is better when w increases. As a matter of fact, the EvaluateProbability() procedure estimates the probability of observing the noisy blinded scalar $\tilde{d}_\ell$ knowing two w-bit word predictions on

[10] without the empirical improvements discussed in Section 3.1

two separate w-bit sections of $\tilde{d}_\ell$. Therefore, if w is too small this estimation is not good enough to distinguish good predictions from wrong ones (Figure 1 illustrates this behaviour).

Our proposal will nevertheless reduce w to its minimum ($w = 1$) and cope with the above mentioned issue by calling EvaluateProbability() (step 12 of Algorithm 1 and step 12 of Algorithm 2) over the two $(w + i - 1)$-bit words $\bar{r}_\ell$ and $\bar{d}_\ell$ instead of the two w-bit words $\hat{r}_\ell$ and $\hat{d}_\ell$.

However, doing this directly has a desastrous effect. On the first iteration of Algorithm 1, many $\tilde{r}_\ell$ are actually wrongly estimated (even if they were the best candidates selected in Algorithm 2) and they remain wrong for the rest of the execution until the end. However, the original implementation deals naturally with them because future probability estimations with future w-bit predictions on these wrong $\tilde{r}_\ell$ quickly decrease to give these wrong starts lower and lower weights in the computation of the best candidate for the bits of d. If we apply our first proposal directly, these wrong starts will keep their high probability estimations for more iterations (since we now involve their successful past in the computation). These wrong starts will then create more chance to choose a wrong candidate for the guessed bit of d. We propose here to solve this problem by loosening the selection procedure of the $\tilde{r}_\ell$.

Keeping a List of the Blinding Factors Best Candidates In a nutshell, the idea is to modify Algorithms 1 and 2 such that instead of working on a single value $\bar{r}_\ell$ (for each $\ell < N$) which is updated bit-by-bit at each iteration (step 14 of Algorithm 2), the algorithms will keep a pool of good estimates for $\tilde{r}_\ell$. Intuitively, if the list of potential candidates is large enough, it will contain the correct value of $\bar{r}_\ell$ for the current iteration. We will see that small list sizes are enough to match and exceed the original algorithm effectiveness.

Algorithms Improvements in Detail Algorithms 3 and 4 describe in detail the full improvements. Concretely, the modifications compared to Algorithms 1 and 2 are threefold:

– the window size w is forced to its minimum ($w = 1$) and then does not appear in the algorithm anymore.
– the list of recovered blinding factors at iteration i, *i.e.* $\{\tilde{r}_\ell\}_{\ell<N}$ where the $\tilde{r}_\ell$ are defined over $i - 1$ bits, is replaced by a 2D array (denoted Lr[11]) of $N \times L$ best candidates for each $\tilde{r}_\ell$. This array is updated at each iteration of Algorithm 4. Note that during the first iterations ($i \leqslant log_2(L)$), all possible candidates are kept until the list is full.
– the probability estimation is done over t msb of $\bar{r}_\ell$ and $\bar{d}_\ell$ instead of the w msb in Algorithm 1. Note that if $t \geqslant R$, then all the bits of $\bar{r}_\ell$ and $\bar{d}_\ell$ are considered in the probability estimation at each iteration.

[11] The array Lr must be initialized to an integer array of dimension $N \times L$ with all cells initialized to -1 but the first column ($Lr[i][0]$ for all $i < N$) which must be initialized to 0.

Together, the last two changes aim at decreasing the value of w to its minimum and therefore reduce the algorithm complexity without damaging too much the algorithm success rate. The overall complexity of steps 1 and 2 becomes then $O(N \times L)$. (More precisely, Step 1 runs $4 \times N \times L$ loop iterations.)

```
Parameter : Iteration $i$
Parameter : Bit error rate $\epsilon_b$
Parameter : Max list size $L$ for the candidate lists of $\tilde{r}_\ell$
Parameter : Window size $t$: this size defines the number of msb to select for probability
            estimations
Input     : $\{\tilde{d}_\ell\}_{\ell<N}$: $N$ noisy scalars
Input     : $Lr$ array of dimension $N \times L$ containing, for each $\ell < N$, the $L$ best
            candidates $\tilde{r}_\ell$
Input     : $d \bmod 2^{i-1}$: $i-1$ lsb of the recovered scalar
Output    : $d \bmod 2^i$
$P \leftarrow$ float 1D array of size 2 initialized with zeros;
// For each possible value of the next bit of the secret scalar;
for $\hat{d} \leftarrow 0$ to $1$ do
    // Prediction of the $i$ lsb of the scalar knowing the first $i-1$ bits;
    $\bar{d} \leftarrow \hat{d} \times 2^{i-1} + d \bmod 2^{i-1}$;
    // For each noisy blinded scalar;
    for $\ell \leftarrow 0$ to $N-1$ do
        // For each possible value of the next bit of the random $r_\ell$;
        for $\hat{r}_\ell \leftarrow 0$ to $1$ do
            // For each $\tilde{r}_\ell$ in the list $Lr[l]$;
            for $s \leftarrow 0$ to $L-1$ do
                $\tilde{r}_\ell \leftarrow Lr[l][s]$;
                if $\tilde{r}_\ell == -1$ then
                    // go to next $\hat{r}_\ell$ value;
                    Break;
                $\bar{r}_\ell \leftarrow \hat{r}_\ell \times 2^{i-1} + \tilde{r}_\ell$;
                // Predict $w + i - 1$ lsb of $d_\ell$;
                $\bar{d}_\ell \leftarrow (\bar{r}_\ell \times E + \bar{d}) \bmod 2^i$;
                // Define $d_\ell^t$, the $t$ msb of $\bar{d}_\ell$;
                $d_\ell^t \leftarrow \left\lfloor \bar{d}_\ell / 2^{max(0,i-t)} \right\rfloor$;
                // Define $r_\ell^t$, the $t$ msb of $\bar{r}_\ell$;
                $r_\ell^t \leftarrow \left\lfloor \bar{r}_\ell / 2^{max(0,i-t)} \right\rfloor$;
                // Compute the probability of observing $\tilde{d}_\ell$, knowing $\hat{d}$ blinded by $r_\ell^t$;
                $p \leftarrow \mathrm{EvaluateProbability}(r_\ell^t, d_\ell^t, \tilde{d}_\ell, max(0, i-t), min(t, i), \epsilon_b)$;
                $P[\hat{d}] \leftarrow P[\hat{d}] + p$;
$d^* \leftarrow \mathrm{argmax}(P) \times 2^{i-1} + d \bmod 2^{i-1}$;
Return    : $d^*$
```

Algorithm 3: Improved Algorithm Step 1

3.4 Simulation Results and Comparisons

We conducted simulations in order to evaluate and compare the new algorithms to the original proposition of [15]. As in Figure 1, the results give the average (over 50 tentatives) number of bits of d guessed correctly before a wrong bit appears, as a function of the number of traces N used for the attack. This

```
Parameter : Iteration $i$
Parameter : Bit error rate $\epsilon_b$
Parameter : Max list size $L$ for the candidate lists of $\tilde{r}_\ell$
Input     : $Lr$ array of dimension $N \times L$ containing, for each $\ell < N$, the $L$ best
            candidates $\tilde{r}_\ell$ on $i-1$ bits
Input     : $d^*$: $i$ lsb of the recovered scalar from Step 1
Output    : Updated $Lr$ array with best candidates $\tilde{r}_\ell$ on $i$ bits
// For each noisy blinded scalar;
for $\ell \leftarrow 0$ to $N-1$ do
    $lr \leftarrow$ number of loaded elements in $Lr[l]$ $(lr \leqslant L)$;
    // Create temporary list $Lr_\ell$ of size $2lr$;
    $Lr_\ell \leftarrow$ integer 1D array of size $2lr$;
    $P \leftarrow$ float 1D array of size $2lr$ initialized with zeros;
    // For each $\tilde{r}_\ell$ in the list $Lr[l]$;
    for $s \leftarrow 0$ to $lr-1$ do
        $\tilde{r}_\ell \leftarrow Lr[l][s]$;
        // Add the two possible values of the next bit of the blinding factor $r_\ell$ to
        the temporary list;
        $Lr_\ell[s] \leftarrow \tilde{r}_\ell$;
        $Lr_\ell[s+lr] \leftarrow 2^{i-1} + \tilde{r}_\ell$;
    if $2lr \leqslant L$ then
        // If $Lr_\ell$ is small enough, keep all $\tilde{r}_\ell$ candidates;
        $Lr[l][0 \cdots 2lr-1] \leftarrow Lr_\ell$;
    else
        // For each $\tilde{r}_\ell$ in the list $Lr_\ell$;
        for $s \leftarrow 0$ to $2lr-1$ do
            $\bar{r}_\ell \leftarrow Lr_\ell[s]$;
            // Predict $i$ lsb of $d_\ell$;
            $\bar{d}_\ell \leftarrow (\bar{r}_\ell \times E + d^*) \bmod 2^i$;
            // Compute the probability of observing $\tilde{d}_\ell$, knowing $d^*$ blinded by $\bar{r}_\ell$;
            $p \leftarrow \mathrm{EvaluateProbability}(\bar{r}_\ell, \bar{d}_\ell, \tilde{d}_\ell, 0, i, \epsilon_b)$;
            $P[s] \leftarrow p$;
        $Lr[l] \leftarrow$ best $L$ candidates in $Lr_\ell$ from their probability estimations $P$;
Return : $Lr$
```
Algorithm 4: Improved Algorithm Step 2

number of correct bits are majored by R since the algorithms studied here stop when the R lsb of d are found. Apart from R and K, various parameters have an impact on the efficiency and the effectivness of the algorithms, notably:

- L: the maximum size of the best candidate pool for the blinding factors $\tilde{r}_\ell$ for each noisy blinded scalar. We recall here that the complexity of Algorithms 3 and 4 increase linearly with L;
- w: the window size, only the original algorithms are affected by w, the complexity of Algorithms 1 and 2 increase exponentially with w;
- t: the number of bits involved in the probability estimation of $\bar{r}_\ell$ and $\bar{d}_\ell$ with respect to $\tilde{d}_\ell$.

Our first simulations are conducted to find the best empirical value for t. Once t is chosen, we will focus on the parameter L and its impact on the effectiveness (compared to the original algorithm when w changes).

Recall that t has no impact on the computational cost of the algorithms, so it can be chosen freely. Figure 2 displays simulation results for the new algorithm with the parameter t taking its values in $\{6, 8, 10, 16, 24, R\}$[12] and small values for L. It appears that $t = 16$ provides better results than greater or smaller values of t in our setup ($K = 256, R = 64$).

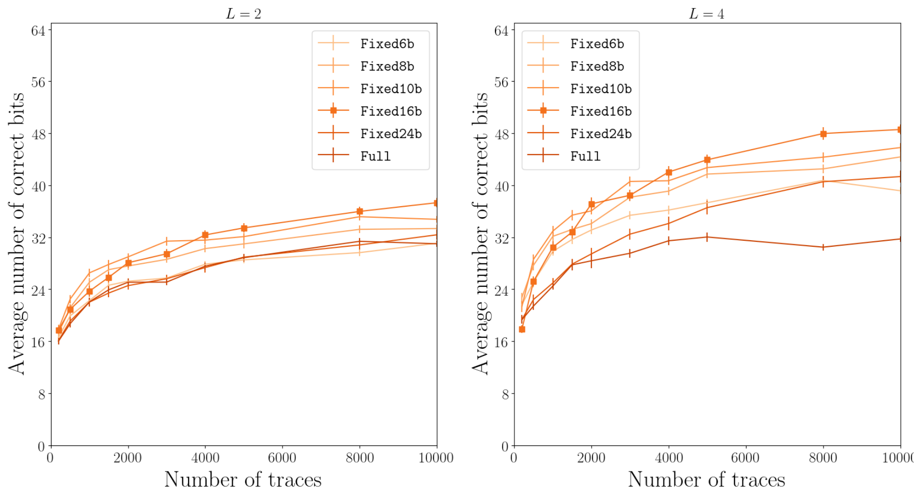

Fig. 2. Simulation Results $K = 256, R = 64, \epsilon_b = 0.15$.

Figure 3 compares the original algorithm for various values of w to the new algorithm (with $t = 16$) for various values of L. From these results, we have equivalent effectivness between the original algorithm with $w = 7$ and the new algorithm with $L = 4$. However, the new algorithm is 2^{10} times more efficient than the original algorithm for these parameters. The gap of efficiency seems to increase with w and L since, for another pair of results ($w = 8$ for the original algorithm and $L = 8$ for the new algorithm) the multiplicative factor between both algorithm complexity is doubled (2^{11}) whereas the new algorithm clearly outperforms the original one. Finally, let us also remark that the new algorithm with $L > 16$ reaches the limit of 64-bit correctly recovered on average (*i.e.* a 100% success rate since 64 is the maximum number of recovered bits) in less than 10000 traces. We recall that these algorithms must reach the end with correct 64-bit lsb of d (since in our simulation we choose $R = 64$) for the overall attack to be successful.

[12] for $t = R$, at iteration i, all bits of $\bar{r}_\ell$ and $\bar{d}_\ell$ are considered for probability estimation, this version is labeled "`Full`"

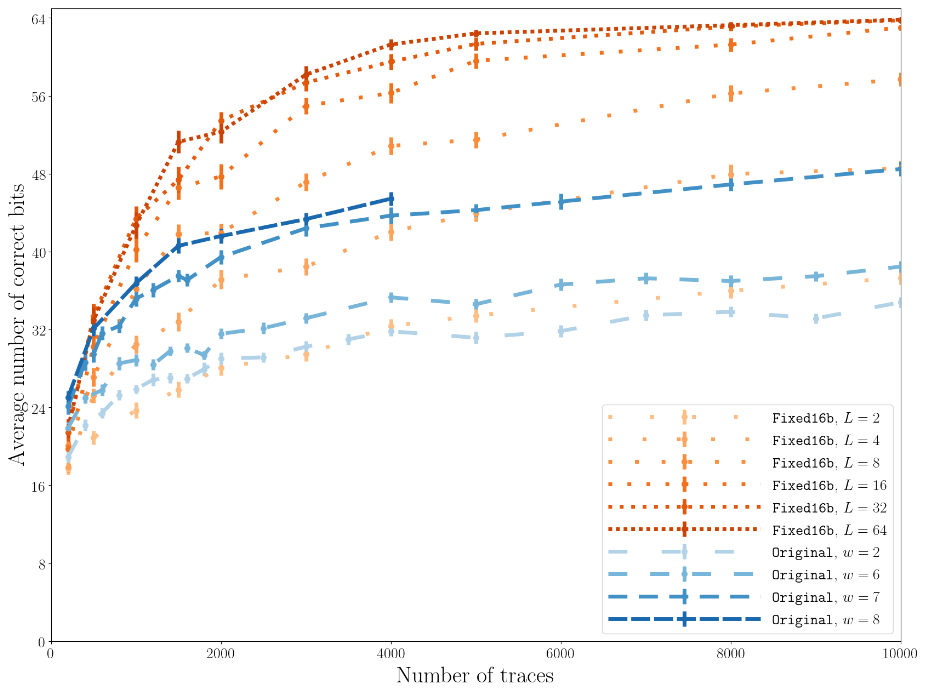

Fig. 3. Simulation Results $K = 256, R = 64, \epsilon_b = 0.15$.

Finally, Figure 4 provides simulation results for $R = 64, 96, 120$ for the new algorithm ($t = 16, L = 32$) and two different bit-error-rate ($\epsilon_b = 0.15$ and $\epsilon_b = 0.13$). These results show, in accordance with original results from [15], that when elliptic curves with structured-order are used, R must be chosen strictly larger $K/2$ in practice for an effective side-channel countermeasure.

4 Conclusion and Future Work

In this paper we exhibited algorithms to recover a secret scalar from many *noisy* blinded scalars (*e.g.* outputs of horizontal side-channel attacks over blinded scalar multiplications) when blinding factors are over more than 32 bits and bit error rate is larger than 10%. Our propositions, both in the general case of random-order elliptic curves as well as in the specific case of structured-order elliptic curves, outperform the best known algorithms for these parameters.

Apart from a series of articles from Schindler *et al.* works on this topic are rather scarse in the literature. This is however a very important aspect of practical side-channel analysis over public-key cryptography and we believe there are still room for improvements. Another interesting avenue for future work is to formulate theoretic bounds on the attacker capability to recover the secret scalar given a set of *noisy* blinded scalars.

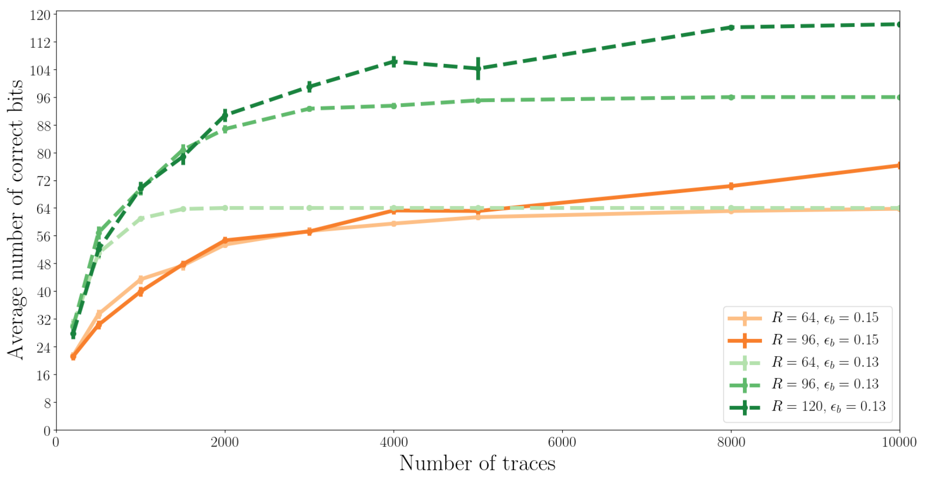

Fig. 4. Simulation $K = 256, L = 32, t = 16$.

Acknowledgments

The authours would like to thank Cyril Bouvier and Bruno Grenet from the ECO group at LIRMM (`https://www.lirmm.fr/eco/`) for their fruitful suggestions and assistance with the cluster MESO@LR (`https://meso-lr.umontpellier.fr/`).

References

1. A. Bauer, É. Jaulmes, E. Prouff, and J. Wild. Horizontal and vertical side-channel attacks against secure RSA implementations. In E. Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.
2. M. Carbone, V. Conin, M. Cornelie, F. Dassance, G. Dufresne, C. Dumas, E. Prouff, and A. Venelli. Deep learning to evaluate secure RSA implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):132–161, 2019.
3. M. Ciet and M. Joye. (virtually) free randomization techniques for elliptic curve cryptography. In S. Qing, D. Gollmann, and J. Zhou, editors, *Information and Communications Security, 5th International Conference, ICICS 2003, Huhehaote, China, October 10-13, 2003, Proceedings*, volume 2836 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2003.
4. J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In **c**C. Ko**c**c and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.
5. B. Feix, M. Roussellet, and A. Venelli. Side-channel analysis on blinded regular scalar multiplications. In W. Meier and D. Mukhopadhyay, editors, *Progress in*

Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings, volume 8885 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2014.

6. FIPS PUB 186-3. *Digital Signature Standard.* National Institute of Standards and Technology, Mar. 2006. Draft.
7. D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer Professional Computing Series, Jan. 2003.
8. J. Heyszl, A. Ibing, S. Mangard, F. D. Santis, and G. Sigl. Clustering algorithms for non-profiled single-execution attacks on exponentiations. In A. Francillon and P. Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 79–93. Springer, 2013.
9. K. Järvinen and J. Balasch. Single-trace side-channel attacks on scalar multiplications with precomputations. In K. Lemke-Rust and M. Tunstall, editors, *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, volume 10146 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2016.
10. E. Nascimento and L. Chmielewski. Applying horizontal clustering side-channel attacks on embedded ECC implementations. In T. Eisenbarth and Y. Teglia, editors, *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, volume 10728 of *Lecture Notes in Computer Science*, pages 213–231. Springer, 2017.
11. E. Nascimento, L. Chmielewski, D. Oswald, and P. Schwabe. Attacking embedded ECC implementations through cmov side channels. In R. Avanzi and H. M. Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, volume 10532 of *Lecture Notes in Computer Science*, pages 99–119. Springer, 2016.
12. G. Perin, L. Imbert, L. Torres, and P. Maurine. Attacking randomized exponentiations using unsupervised learning. In E. Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*, pages 144–160. Springer, 2014.
13. W. Schindler and K. Itoh. Exponent blinding does not always lift (partial) SPA resistance to higher-level security. In J. López and G. Tsudik, editors, *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, volume 6715 of *Lecture Notes in Computer Science*, pages 73–90, 2011.
14. W. Schindler and A. Wiemers. Power attacks in the presence of exponent blinding. *J. Cryptographic Engineering*, 4(4):213–236, 2014.
15. W. Schindler and A. Wiemers. Efficient Side-Channel Attacks on Scalar Blinding on Elliptic Curves with Special Structure. *NIST Workshop on ECC Standards*, 2015.
16. W. Schindler and A. Wiemers. Generic power attacks on RSA with CRT and exponent blinding: new results. *J. Cryptographic Engineering*, 7(4):255–272, 2017.
17. R. Specht, J. Heyszl, M. Kleinsteuber, and G. Sigl. Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In S. Mangard and A. Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International*

Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2015.

18. Standards for Efficient Cryptography Group (SECG). *SEC 2 : Recommended Elliptic Curve Domain Parameters.* Certicom Research, 2000.
19. L. Weissbart, S. Picek, and L. Batina. One trace is all it takes: Machine learning-based side-channel attack on eddsa. Cryptology ePrint Archive, Report 2019/358, 2019. `https://eprint.iacr.org/2019/358`.

A Brief Analysis of the Final Vertical Side-channel Analysis (Attack Step 3)

Assuming that the attacker went successfully through the overall attack path described in Section 1.2, we analyse here the success rate of the last step of the attack: the vertical side-channel attack given a set of noisy blinded scalars (or exponents in the case of RSA) $\{\tilde{d_l}\}_{l<N}$ and their corresponding blinding factors $\{r_l\}_{l<N}$. A simple attack algorithm follows:

Set a small window size w (*e.g.* $w = 4$), the vertical attack will find each bit of the secret scalar (resp. exponent) d iteratively with predictions over w consecutive bits. In the case of ECC, where the order of the elliptic curve E is public, at iteration i the attacker possesses $i-1$ bits of d (as well as the blinding factors $\{r_l\}_{l<N}$). Then, for each of the possible next w bits of d (denoted $\hat{d}^w$, which takes values in $\mathbb{F}_{2^w}$), the attacker can predict the ith to $(i+w-1)$th bit of the N real blinded scalars. The attacker can then select the hypothesis on $\hat{d}^w$ that predicts the best[13] the observations $\{\tilde{d_l}\}_{l<N}$ and only keep the first bit of it ($\hat{d}^w$ mod 2 which becomes the ith bit of d) and then move to the next iteration. The overall attack complexity is $O(2^w NK)$.

In the case of RSA, the value of $\phi(n)$ (or $\phi(p)$ for a CRT implementation) is unkwown, the attack will then process similarly but guessing the bits of $\phi(n)$ (resp. $\phi(p)$) simultaneously with the bits of d. Hence, at each iteration, the attacker will go through all values of the next w bits of d as well as the next w bits of $\phi(n)$ (resp. $\phi(p)$) and, for each of them, predict w bits of the N blinded exponents and confront them to w bits of the N noisy blinded exponent. The overall attack complexity is $O(2^{2w}NK)$.

Figure 5 shows the average number of correctly guessed bits of d for ECC ($K = 256$) and RSA ($K = 1024$) and for various values of $w \in \{1, 2, 4\}$. We conducted these simulations for three bit error rates, up to $\epsilon_b = 0.3$, which is larger than any bit error rate considered in this paper. These results confirm the claim that few tens of noisy blinded scalars (resp. exponents), given their blinding factors, are enough to find the secret scalar (resp. exponent).

[13] in our simulations we simply used the majority rule: the number of matching bits between the predictions and the noisy blinded scalars. This step will perform even better by using the side-channel traces in a more classical side-channel attack.

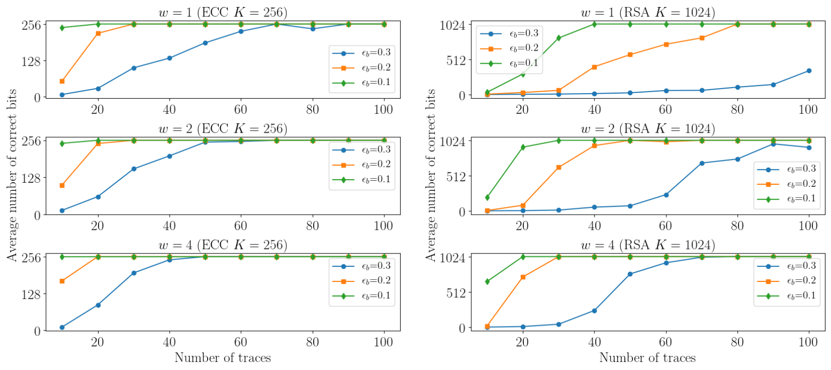

Fig. 5. Average number of correct bits of d (over 20 trials).