Side-Channel Attacks on Blinded Scalar Multiplications Revisited

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

[1] NinjaLab, Montpellier, France
thomas@ninjalab.io
[2] LIRMM, CNRS,
University of Montpellier, Montpellier, France
https://ninjalab.io

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 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 [2,7–11,16,18]. These attacks do not usually recover

S. Belaïd and T. Güneysu (Eds.): CARDIS 2019, LNCS 11833, pp. 95–108, 2020.
https://doi.org/10.1007/978-3-030-42068-0_6

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 on so-called structured-order elliptic curves (very common in Elliptic Curves Cryptography). 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 [12] 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 [13], applied to specific elliptic curves [5,14] and to RSA with CRT [15]. 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 $\{T_\ell\}_{\ell<N}$[1].

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.

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

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[2] (see *e.g.* [1,2,18]) 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.* [7–11,16]), 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.

Step 3: The last step of the attack recovers the secret scalar d from $\{\tilde{d}_\ell\}_{\ell\in J}$ and $\{r_\ell\}_{\ell\in J}$. A powerfull *vertical* side-channel attack can be mounted on the remaining traces. Such an attack is described in [5].

1.3 Paper Organization and Contributions

This work focuses on improvements in *Step 2* in the specific case of elliptic curves whose order is close to a power of 2. Section 2 describes the previous works, namely the best known attack in this setting [14]. Our strategy and results are presented in Sect. 3. Our simulations show significant increases in the success rates for blinding factors up to $R = K/2$ compared to [14].

2 Previous Works

In [14], 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 [17], NIST curves [6].

[2] 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.

2.1 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 [14]) propose an efficient algorithm to recover the secret d that comprises two 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 and recover the full secret scalar d.

Phase 2 corresponds to step 3 of our overall attack scheme described in Sect. 1.2 and, as observed by the authors of [14], is not the critical path of the attack. In other words, if Phase 1 is successful (*i.e.* the R lsb of d are correctly found) then Phase 2 will results in recovering the full value of d with high probability.

2.2 Schindler and Wiemers' Phase 1 Algorithm

It is described in [14, 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 Algorithms 1 and 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 ϵ_b
Input : $\{\tilde{d}_\ell\}_{\ell<N}$: N noisy scalars
Input : $\{\bar{r}_\ell\}_{\ell<N}$: $i-1$ lsb of the recovered blinding factors
Input : $d \bmod 2^{i-1}$: $i-1$ lsb of the recovered scalar
Output : d^*: best guess for $d \bmod 2^{w+i-1}$

$P \leftarrow$ float 1D array of size 2^w initialized with zeros;
// For each possible value of the next w bits of the secret scalar;
for $\hat{d} \leftarrow 0$ **to** $2^w - 1$ **do**
  // Prediction of the $w+i-1$ 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 w bits of the random r_ℓ;
    for $\hat{r}_\ell \leftarrow 0$ **to** $2^w - 1$ **do**
      $\bar{r}_\ell \leftarrow \hat{r}_\ell \times 2^{i-1} + \tilde{r}_\ell$;
      // Predict $w+i-1$ lsb of d_ℓ;
      $\bar{d}_\ell \leftarrow (\bar{r}_\ell \times E + \bar{d}) \bmod 2^{w+i-1}$;
      // Define $\hat{d}_\ell$, the w msb of $\bar{d}_\ell$;
      $\hat{d}_\ell \leftarrow \lfloor \bar{d}_\ell / 2^{i-1} \rfloor$;
      // Compute the probability of observing $\tilde{d}_\ell$, knowing $\hat{d}$ blinded by $\hat{r}_\ell$;
      $p \leftarrow \text{EvaluateProbability}(\hat{r}_\ell, \hat{d}_\ell, \tilde{d}_\ell, i, w, \epsilon_b)$;
      $P[\hat{d}] \leftarrow P[\hat{d}] + p$;
$d^* \leftarrow \text{argmax}(P) \times 2^{i-1} + (d \bmod 2^{i-1})$;
Return : d^*

Algorithm 1. Phase 1, Step 1 of [14, 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 effectiveness. We will briefly present them in the next section. However, since these improvements are based on hand-picked thresholds by the authors of [14] 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.

2.3 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 [14] 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 ϵ_b
Input : $\{\bar{d}_\ell\}_{\ell<N}$: 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_ℓ;
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_ℓ;
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 [14, 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, hand-picked numbers of noisy blinded scalars to be used in Step 1 for various bit error rates ϵ_b and iteration numbers 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.

2.4 Some Results

It is shown in [14] 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 [14], 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 of Algorithms 1 and 2 effectivness for various values of w. 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` [17] (aka the Bitcoin's curve).

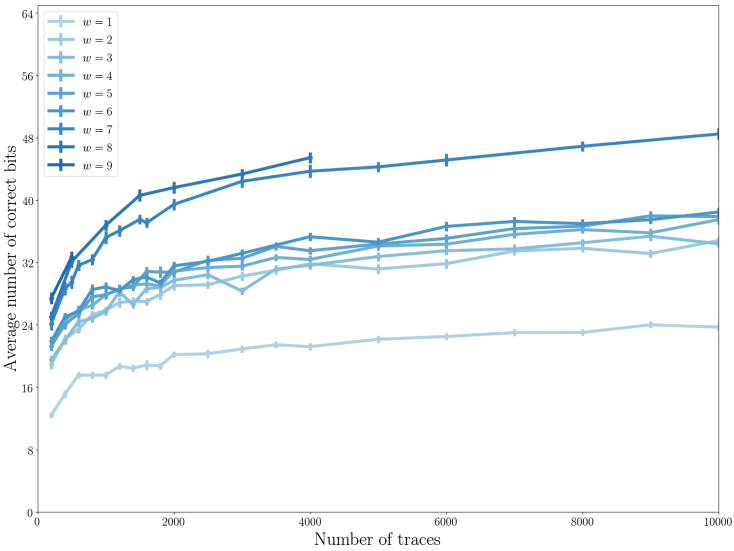

Fig. 1. Simulations for $K = 256, R = 64, \epsilon_b = 0.15$ on curve `secp-256-k1`.

In Fig. 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 [14] are scattered over a few parameters. We believe that Fig. 1 provides a complementary point of view on the efficiency of the correction algorithm of [14][3]. 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 Improved Algorithms

3.1 First Observations

As remarked earlier (and in [14]) 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) improves as 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 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 (Fig. 1 illustrates this behaviour).

[3] Without the empirical improvements discussed in Sect. 2.3.

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 $\tilde{r}_\ell$.

3.2 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.

3.3 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[4]) 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.

[4] 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 ϵ_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_ℓ;
    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_ℓ;
        $\bar{d}_\ell \leftarrow (\bar{r}_\ell \times E + \bar{d}) \bmod 2^i$;
        // Define d_ℓ^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_ℓ^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_ℓ^t;
        $p \leftarrow \text{EvaluateProbability}(r_\ell^t, d_\ell^t, \tilde{d}_\ell, max(0, i-t), min(t, i), \epsilon_b)$;
        $P[\hat{d}] \leftarrow P[\hat{d}] + p$;
$d^* \leftarrow \text{argmax}(P) \times 2^{i-1} + d \bmod 2^{i-1}$;
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 [14]. As in Fig. 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 number of

Parameter : Iteration i
Parameter : Bit error rate ϵ_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 ℓ ← 0 to N − 1 do
    lr ← number of loaded elements in Lr[l] (lr ⩽ L);
    // Create temporary list Lr_ℓ of size 2lr;
    Lr_ℓ ← integer 1D array of size 2lr;
    P ← float 1D array of size 2lr initialized with zeros;
    // For each r̃_ℓ in the list Lr[l];
    for s ← 0 to lr − 1 do
        r̃_ℓ ← Lr[l][s];
        // Add the two possible values of the next bit of the blinding factor r_ℓ to
        the temporary list;
        Lr_ℓ[s] ← r̃_ℓ;
        Lr_ℓ[s + lr] ← 2^(i−1) + r̃_ℓ;
    if 2lr ⩽ L then
        // If Lr_ℓ is small enough, keep all r̃_ℓ candidates;
        Lr[l][0 ··· 2lr − 1] ← Lr_ℓ;
    else
        // For each r̃_ℓ in the list Lr_ℓ;
        for s ← 0 to 2lr − 1 do
            r̄_ℓ ← Lr_ℓ[s];
            // Predict i lsb of d_ℓ;
            d̄_ℓ ← (r̄_ℓ × E + d*) mod 2^i;
            // Compute the probability of observing d̄_ℓ, knowing d* blinded by r̄_ℓ;
            p ← EvaluateProbability(r̄_ℓ, d̄_ℓ, d̃_ℓ, 0, i, ε_b);
            P[s] ← p;
        Lr[l] ← best L candidates in Lr_ℓ from their probability estimations P;
Return : Lr
```

Algorithm 4. Improved Algorithm Step 2

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 $\{6, 8, 10, 16, 24, R\}$[5] and small values for L. It appears that $t = 16$ provides better results than greater or smaller values of t in our setup ($K = 256, R = 64$).

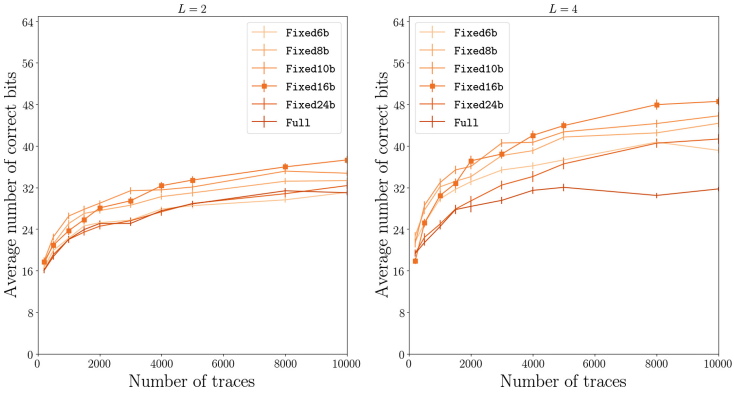

Fig. 2. Simulation results $K = 256, R = 64, \epsilon_b = 0.15$.

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.

Finally, Fig. 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 [14], 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.

[5] 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`".

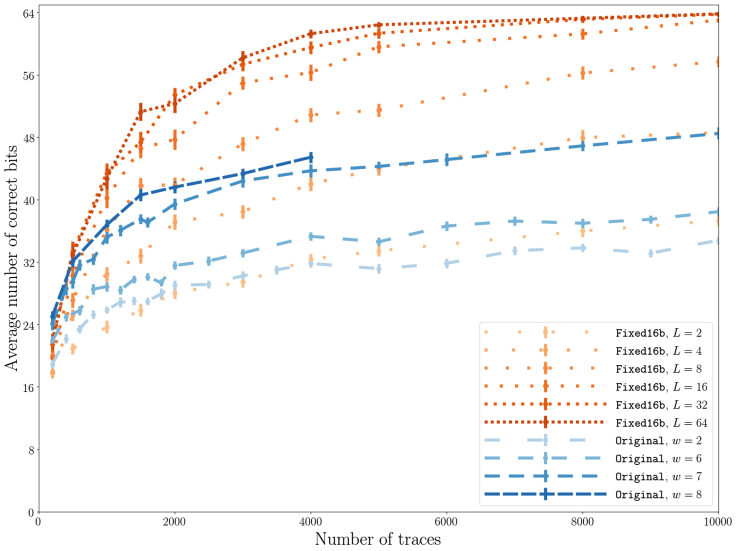

Fig. 3. Simulation results $K = 256, R = 64, \epsilon_b = 0.15$.

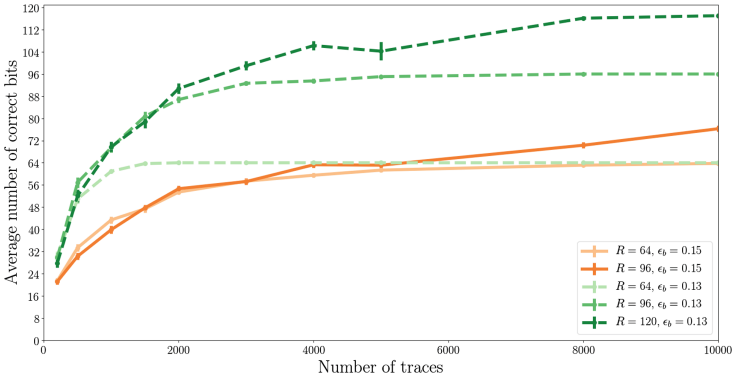

Fig. 4. Simulation $K = 256, L = 32, t = 16$.

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 large and bit error rate is larger than 10%. Our propositions, 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.

Acknowledgments. The authors 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. Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal and vertical side-channel attacks against secure RSA implementations. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 1–17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36095-4_1
2. Carbone, M., et al.: Deep learning to evaluate secure RSA implementations. IACR Trans. Cryptograph. Hardw. Embed. Syst. **2019**(2), 132–161 (2019). https://doi.org/10.13154/tches.v2019.i2.132-161
3. Ciet, M., Joye, M.: (Virtually) free randomization techniques for elliptic curve cryptography. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 348–359. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39927-8_32
4. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koc, **c**C.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48059-5_25
5. Feix, B., Roussellet, M., Venelli, A.: Side-channel analysis on blinded regular scalar multiplications. In: Meier, W., Mukhopadhyay, D. (eds.) INDOCRYPT 2014. LNCS, vol. 8885, pp. 3–20. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13039-2_1
6. FIPS PUB 186–3: Digital Signature Standard. National Institute of Standards and Technology, March 2006. Draft
7. Heyszl, J., Ibing, A., Mangard, S., De Santis, F., Sigl, G.: Clustering algorithms for non-profiled single-execution attacks on exponentiations. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 79–93. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08302-5_6
8. Järvinen, K., Balasch, J.: Single-trace side-channel attacks on scalar multiplications with precomputations. In: Lemke-Rust, K., Tunstall, M. (eds.) CARDIS 2016. LNCS, vol. 10146, pp. 137–155. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54669-8_9
9. Nascimento, E., Chmielewski, Ł.: Applying horizontal clustering side-channel attacks on embedded ECC implementations. In: Eisenbarth, T., Teglia, Y. (eds.) CARDIS 2017. LNCS, vol. 10728, pp. 213–231. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75208-2_13
10. Nascimento, E., Chmielewski, Ł., Oswald, D., Schwabe, P.: Attacking embedded ECC implementations through cmov side channels. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, vol. 10532, pp. 99–119. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69453-5_6

11. Perin, G., Imbert, L., Torres, L., Maurine, P.: Attacking randomized exponentiations using unsupervised learning. In: Prouff, E. (ed.) COSADE 2014. LNCS, vol. 8622, pp. 144–160. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10175-0_11
12. Schindler, W., Itoh, K.: Exponent blinding does not always lift (partial) spa resistance to higher-level security. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 73–90. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21554-4_5
13. Schindler, W., Wiemers, A.: Power attacks in the presence of exponent blinding. J. Cryptograph. Eng. **4**(4), 213–236 (2014). https://doi.org/10.1007/s13389-014-0081-y
14. Schindler, W., Wiemers, A.: Efficient Side-Channel Attacks on Scalar Blinding on Elliptic Curves with Special Structure. In: NIST Workshop on ECC Standards (2015)
15. Schindler, W., Wiemers, A.: Generic power attacks on RSA with CRT and exponent blinding: new results. J. Cryptograph. Eng. **7**(4), 255–272 (2017). https://doi.org/10.1007/s13389-016-0146-1
16. Specht, R., Heyszl, J., Kleinsteuber, M., Sigl, G.: Improving Non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In: Mangard, S., Poschmann, A.Y. (eds.) COSADE 2014. LNCS, vol. 9064, pp. 3–19. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21476-4_1
17. Standards for Efficient Cryptography Group (SECG): SEC 2: recommended elliptic curve domain parameters. Certicom Research (2000). http://www.secg.org/collateral/sec2_final.pdf
18. Weissbart, L., Picek, S., Batina, L.: 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