Exponent Blinding Does Not Always Lift (Partial) Spa Resistance to Higher-Level Security

Werner Schindler[1] and Kouichi Itoh[2]

[1] Bundesamt für Sicherheit in der Informationstechnik (BSI)
Godesberger Allee 185–189, 53175 Bonn, Germany
Werner.Schindler@bsi.bund.de
[2] Fujitsu Laboratories Ltd., 211–8588, 4-1-1, KamiKodanaka, Nakahara-ku, Kawasaki, Japan
kito@labs.fujitsu.com

Abstract. Exponent blinding is known as a secure countermeasure against side-channel attacks. If single power traces reveal some exponent bits, an attack by Fouque et al. applies that recovers the exponent. However, this attack becomes infeasible if some of the guessed bits are incorrect. Thus, the attack was not assumed to be a realistic threat. In this paper we present two variants of a novel generic attack, which works for considerable error rates at each bit position, disproving the hypothesis that mere exponent blinding is always sufficient. We confirmed experimentally that our attack permits up to 28% (RSA case) or 23% (ECC case) error bits.

1 Introduction

Blinding mechanisms such as base and exponent blinding [8] have been effective algorithmic countermeasures against side-channel attacks. Both seem to prevent pure global timing attacks [9], since no pure timing attack is presently known defeating either base or exponent blinding.

However, base blinding does not protect against local timing attacks [11,1], which exploit timing information on the particular elementary operations: modular multiplications and squarings in the RSA case, point doubling and point addition (or arithmetic sub-operations thereof) in the ECC case. Attacks on table-based RSA implementations [11,1] exploit the pre-computation phase of modular exponentiation and the fact that the i^{th} elementary operation in the exponentiation phase is always of the same type (either squaring or a multiplication by a particular table value).

A cryptographic device must be also secure against power analysis [9], which in particular comprises Simple Power Analysis (SPA) and Differential Power Analysis (DPA). However, exponent blinding prevents the alignment of power traces corresponding to elementary operations of the same type. This shall prevent an attacker from combining information from several RSA exponentiations/ECC

J. Lopez and G. Tsudik (Eds.): ACNS 2011, LNCS 6715, pp. 73–90, 2011.
© Springer-Verlag Berlin Heidelberg 2011

scalar point multiplications. It might thus be assumed that exponent blinding protects against power attacks if the device is resistant against SPA attack, or more generally, against attacks on single exponentiations/scalar point multiplications. The assumption that exponent blinding automatically lifts SPA or partial SPA resistance (i.e. the device is secure enough to protect some private key bits against SPA) to higher-level security assertions (e.g., DPA resistance) should be valid in many cases of practical relevance. However, we present a new generic attack that shows that this conclusion is not true in general. We assume that the targeted device applies the square and always multiply (S&aM) algorithm [4], resp. a double and always add (D&aA) algorithm, as an SPA resistance algorithm. At first we sketch related previous results.

Applying exponent blinding is a powerful solution even when the device is partially SPA-resistant. In theory, a device with partial SPA resistance that only applies exponent blinding is not secure against side-channel attacks [2]. However this attack assumes a strong assumption: *all observed bits* of the randomized exponent must be perfectly revealed from a single power trace *with no error bits*, which is not easy to achieve in real environment because noise is included. There are some known attacks on the S&aM algorithm. First is the address-bit power analysis technique [6] because the address differs between the real and dummy operations. However, large hardware multipliers cause strong noise, which should hide the power consumption of the memory addresses. Second attack is a chosen message technique using $c = -1 \pmod N$ proposed by Yen et al [12]. Itoh et al.'s result [7] shows clear waveforms using auto-correlation techniques. However, filtering the messages, checking whether $c = -1 \pmod N$, prevents this attack. If $c = -1 \pmod N$ simply the lowest bit of the exponent is used to calculate an output. Third is an attack proposed by Courrège et al. [5] that utilizes the power consumption of partial multiplications on word length. It is effective if messages can be chosen, which minimize partial hamming weights. However, S&aM reduces its effectivity because the attacker can use only $c = -1 \pmod N$, which is easily prevented by the filtering. That is, we do not know concrete attacks, which fulfil the above assumption in real-world environments when a simple filtering technique is used.

Can the attacker reveal the private key even when the above assumption, consolidated knowledge on some key bits, is not satisfied? The scenario is the following: the attacker guesses the blinded exponent bits on basis of the corresponding waveforms. Some of the guessed bits may be wrong but the attacker does not know their positions. This scenario is realistic if the waveforms of real and dummy operations are similar, with noisy power traces. If there are 5% unnoticed error bits a brute-force attack costs $\approx \sum_{j=0}^{51} \binom{1024}{j} = 2^{288}$ steps for 1024-bit RSA, or $\approx \sum_{j=0}^{13} \binom{256}{j} = 2^{71}$ steps for 256-bit ECC.

This motivates the following idea: if the operations cannot perfectly be distinguished, the device should be secure against side-channel attack by asserting exponent blinding and input filtering. However, our attack shows this idea does not suffice in general. We show that the attacker is able to identify exponents with identical exponent blinding factors (basic attack) or sums of exponents with

identical sums of exponent blinding factors (enhanced attack), respectively, undermining the declared goal of exponent blinding. The basic attack searches for t exponents with identical exponent blinding factors, which allows to simply apply the 'best of t' rule (majority decision). The enhanced attack applies more sophisticated techniques. Both variants tolerate if all exponent bit positions allow only uncertain guesses. In our analysis we assume identical error rate ϵ_b for all bit positions and traces. Interestingly, even additionally applied base blinding might not prevent our attack. Fortunately, generic countermeasures exist.

We mention that reference [3] also considers the recovery of a secret RSA key from defective information. It is a theoretical attack that tolerates certain errors rates in the binary representations of the particular components of (p, q, d, d_p, d_q), (p, q, d), or (p, q), respectively. Results and techniques from [3] cannot be transferred to our situation where an attacker does not get any direct information on the primes p and q. Our attack works against RSA (with and without CRT) and ECC without any restrictions on the private key nor on the public key.

In Section 2 and Section 3 we describe the basic version and an enhanced version of our new attack and present experimental results. Unlike the basic attack, the enhanced attack needs only a small number of power traces to find the private key and cannot effectively be prevented by limiting the number of operations with the targeted key. Instead, large blinding factors are necessary.

2 Basic Attack

The setting of the attack is as follows. A target device (typically a smart card) executes RSA or ECC operations. In both the basic and enhanced attacks we assume that S & aM (RSA)/ D& aA (ECC) are used within exponent/scalar blinding. The attacker performs a power attack on the target device. From single power traces he derives information on the corresponding exponentiations or scalar point multiplications with the secret key.

2.1 Notation

Our attack is generic and applies identically to RSA with CRT, RSA without CRT, and ECC. To avoid clumsy formulations we will simply speak of 'multiplication', 'square' and 'exponent' in the following, which correspond to 'addition', 'doubling' and 'scalar' in the ECC context. The blinded exponents are

$$v_j := d + r_j y \qquad \text{for } j = 1, 2, \ldots \tag{1}$$

where d denotes a long term key. For RSA d is the secret exponent, resp. the secret exponent $(\bmod p)$ (usually denoted by d_p in this context) if the CRT is used. For ECC d equals the secret scalar. Further, $y = \phi(n)$, $y = \phi(p)$, or y equals the order of an elliptic curve, respectively. The integer r_j is the exponent blinding factor used for exponentiation j.

Remark 1. We point out that for RSA with CRT it suffices to recover d_p if at least one (plaintext / ciphertext) pair (x, y) is known, e.g. a digital signature. Then $\gcd(y - x^{d_p} (\bmod n), n) = p$ factors the modulus n.

We assume $d, y < 2^k$ where the integers d and y (resp. only d for ECC applications) are selected randomly. Hence we may assume $\log_2(d), \log_2(y) \approx k$. The blinding factor r_j assumes values in $\{0, 1, \ldots, 2^R - 1\}$. The binary representation of v_j is $(v_{j;k+R-1}, \ldots, v_{j;0})_2$ where leading zero digits are allowed.

On basis of the SPA information provided by power trace j the attacker guesses the randomized exponent with SPA to obtain $\widetilde{v}_j = (\widetilde{v}_{j;n+R-1}, \ldots, \widetilde{v}_{j;0})_2$, which will usually differ from the correct binary representation $(v_{j;k+R-1}, \ldots, v_{j;0})_2$ of v_j. (Otherwise, the j^{th} power trace alone would suffice for a successful SPA.)

The attacker may commit two types of guessing errors: Although $v_{j;i} = 0$ he might guess $\widetilde{v}_{j;i} = 1$ ('10-error'), or despite of $v_{j;i} = 1$ he might guess $\widetilde{v}_{j;i} = 0$ ('01-error'). We denote the integer that corresponds to $(\widetilde{v}_{j;n+R-1}, \ldots, \widetilde{v}_{j;0})_2$ by $\widetilde{v}_j$. Using this notation the attacker guesses

$$\widetilde{v}_j := (d + r_j y) \oplus e_j = v_j \oplus e_j \qquad \text{for } j = 1, 2, \ldots \tag{2}$$

where '$\oplus$' denotes the bitwise XOR operation while the integer e_j expresses the guessing error for exponent v_j. For $j \neq m$ we consider the hamming weight of

$$\widetilde{v}_j \oplus \widetilde{v}_m = (v_j \oplus v_m) \oplus (e_j \oplus e_m). \tag{3}$$

We use the relationship

$$\mathrm{ham}(\widetilde{v}_j \oplus \widetilde{v}_m) = \begin{cases} \mathrm{ham}(e_j \oplus e_m) & (\text{if } r_j = r_m) \\ \mathrm{ham}(v_j \oplus v_m \oplus e_j \oplus e_m) & (\text{if } r_j \neq r_m) \end{cases} \tag{4}$$

to distinguish the cases $r_j = r_m$ and $r_j \neq r_m$.

2.2 Entire Process of the Basic Attack

The attack consists of two phases. In Phase 1 the attacker guesses $\widetilde{v}_1, \widetilde{v}_2, \ldots$ on basis of the corresponding power traces (independent SPA attacks). The goal of Phase 1 is to group the guessed exponents to classes with identical (though unknown) blinding factors ('blinding classes'). Phase 2 considers only the class $\mathcal{C}_w$, which at first contains t elements with (presumably) identical blinding factors('winning class'). The generic approach clearly is to apply the 'majority decision' to each elementary operation guess, based on the guessed binary representations $(\widetilde{v}_{j;n+R-1}, \ldots, \widetilde{v}_{j;0})_2$ for all $\widetilde{v}_j \in \mathcal{C}_w$. We point out that the attacker does not know d, y, r (RSA case), resp. d, r (ECC case) for any blinding class. Pseudoalgorithm 1 describes our attack in more detail. The integer t should be selected with regard to the error probabilities (cf. below). Typically, t is an odd number ($t = 3, 5, \ldots$), cf. Table 2 and Table 3. The attacker decides that $\widetilde{v}_j$ belongs to class $\mathcal{C}_m$ iff $\mathrm{ham}(\widetilde{v}_j \oplus \widetilde{v}) < \gamma$ (a suitably selected threshold) for all $\widetilde{v} \in \mathcal{C}_m$. The algorithm stops when the some class $\mathcal{C}_w$ contains t elements (a so-called 't-birthday').

Pseudoalgorithm 1. (The basic attack)

```
   Select an integer t ≥ 3 and a suitable threshold γ > 0
1) Phase 1: Find a class C_m with t elements ( = 'winning class')
   j:=1; s:=0; stop:=0;
   while (stop = 0) {
     attacker estimates ṽ_j;
     if (j = 1) then { s := 1; C_1 := {ṽ_1}; }
     else { m:=1; found:=0;
       while ( (m ≤ s) && (found = 0) ) {
         if ( ham((ṽ_j ⊕ ṽ)) < γ ) for all ṽ ∈ C_m then {
            C_m := C_m ∪ {ṽ_j} ; found:=1;
             if ( |C_m| = t ) then { w:=m; stop:=1; }
              /* t-birthday, winning class found */ } }
       if (found = 0) then { s^{++}; C_s := {ṽ_j}; }
   } }
2) Phase 2: Apply the majority decision to the elements
            of the winning class C_w
```

Theoretical Background of Phase 1. We show how to distinguish $r_j = r_m$ or $r_j \neq r_m$. This is based on Eqn. (4) which reads

$$\mathrm{ham}(\widetilde{v}_j \oplus \widetilde{v}_m) = \sum_{i=0,1,\ldots,n+R-1} (v_{j;i} \oplus v_{m;i} \oplus e_{j;i} \oplus e_{m;i}) \ , \tag{5}$$

$$\text{which simplifies to} \quad \sum_{i=0,1,\ldots,n+R-1} (e_{j;i} \oplus e_{m;i}) \qquad \text{if } r_j = r_m \tag{6}$$

where $e_{j;i}, e_{m;i}$ are the i^{th} elements of error vectors $e_j = (e_{j;n+R-1}, \ldots, e_{j;0})_2$ and $e_m = (e_{m;n+R-1}, \ldots, e_{m;0})_2$. Let ϵ_b be the error rate in Phase 1, i.e. $e_{j;i}$ or $e_{m;i}$ is '1' with probability ϵ_b, and is '0' with probability $1 - \epsilon_b$. Hence $e_{j;i} \oplus e_{m;i} = 1$ is satisfied with probability $2\epsilon_b(1-\epsilon_b)$. If $r_i = r_m$ we assume that $\widetilde{v}_j \oplus \widetilde{v}_m$ and $\mathrm{ham}(\widetilde{v}_j \oplus \widetilde{v}_m)$ are values that are taken on by random variables X' and $Y' := \mathrm{ham}(X')$, where the components of X' are independent and binomially $B(2\epsilon_b(1-\epsilon_b), 1)$-distributed. Hence Y' is binomially $B(2\epsilon_b(1-\epsilon_b), n+R)$-distributed and thus has mean $\mu_{X'} = 2\epsilon_b(1-\epsilon_b)(n+R)$ and standard deviation $\sigma_{Y'} = \sqrt{Var(\mathrm{ham}(X'))} = \sqrt{2\epsilon_b(1-\epsilon_b)(\epsilon_b^2 + (1-\epsilon_b)^2)(n+R)}$.

In case $r_j \neq r_m$, we can assume that $v_{j;i} \oplus v_{m;i} = 1$ holds with probability $1/2$. Since $(v_{j;i}, v_{m;i})$ and $(e_{j;i}, e_{m;i})$ are independent $v_{j;i} \oplus v_{m;i} \oplus e_{j;i} \oplus e_{m;i} = 0$ is satisfied with probability $1/2$. Therefore, $\widetilde{v}_j \oplus \widetilde{v}_m$ and $\mathrm{ham}(\widetilde{v}_j \oplus \widetilde{v}_m)$ are assumed to be values that are taken on by random variables X and $Y := \mathrm{ham}(X)$, where the components of X are independent and binomially $B(1/2, 1)$-distributed. Hence Y is binomially $B(1/2, n+R)$-distributed and thus has mean $\mu_X = 0.5(n+R)$ and standard deviation $\sigma_Y = \sqrt{Var(\mathrm{ham}(X))} = \sqrt{(n+R)/4}$. Table 2 illuminates the decision step ($R = 16$, $\gamma = \mu_Y - 4\sigma_Y$, sample size = 10000 for each parameter set). The column 'success rate' provides the empirical probability for the decision $r_j = r_m$ when $r_j = r_m$ is indeed true whereas 'miss ratio' shows the empirical probability for $r_j = r_m$ when $r_j \neq r_m$ is true. An appropriate choice of γ, which should consider R is important for the success of the attack. Lowering

Table 1. ham(·) distributions for different sets of parameters

			$\text{ham}(\widetilde{v}_j \oplus \widetilde{v}_m)$							$\text{ham}(\widetilde{v}_j \oplus \widetilde{v}_m)$			
			$r_j = r_m$		$r_j \neq r_m$					$r_j = r_m$		$r_j \neq r_m$	
k	R	ϵ_b	$\mu_{Y'}$	$\sigma_{Y'}$	μ_Y	σ_Y	k	R	ϵ_b	$\mu_{Y'}$	$\sigma_{Y'}$	μ_Y	σ_Y
256	16	0.10	48.96	6.34	136	8.25	1024	16	0.15	265.2	14.06	520	16.12
256	16	0.15	69.36	7.19	136	8.25	1024	16	0.20	332.80	15.04	520	16.12
256	16	0.20	87.04	7.69	136	8.25	1024	16	0.25	390.0	15.61	520	16.12
256	16	0.25	102.00	7.98	136	8.25	1024	16	0.30	436.80	15.92	520	16.12

Table 2. Experimental results for $\gamma = \mu_Y - 4\sigma_Y$

			$r_j = r_m$					$r_j = r_m$	
k	R	ϵ_b	Success ratio	Miss ratio	k	R	ϵ_b	Success ratio	Miss ratio
256	16	0.10	10000/10000	0/10000	1024	16	0.15	10000/10000	1/10000
256	16	0.15	10000/10000	0/10000	1024	16	0.20	10000/10000	0/10000
256	16	0.20	9762/10000	1/10000	1024	16	0.25	10000/10000	1/10000
256	16	0.22	8793/10000	2/10000	1024	16	0.30	8576/10000	0/10000

γ reduces the probability that $\widetilde{v}_j$ is erroneously assumed to belong to blinding class $\mathcal{C}_m$ while the probability for a false rejection $\widetilde{v}_j \notin \mathcal{C}_m$ (and opening a new blinding class) increases. Increasing γ clearly has opposite effects.

Discussion and Experimental Result of Phase 2. Let r_w^* denote the (unknown) common blinding factor of all elements in the winning blinding class $\mathcal{C}_w$. In Phase 2 the majority decision rule yields a false guess for the i^{th} elementary operation of the blinded exponent $v_w^* := d + r_w^* y$ if the majority of the respective error bits $e_{\cdot,i}$ in $\mathcal{C}_w$ is 1. This occurs with probability

$$q_{t,\epsilon_b} := \sum_{s=u+1}^{2u+1} \binom{t}{s} \epsilon_b^s (1-\epsilon_b)^{t-s} \quad \text{for } t = 2u+1\,,$$

and we expect $e_{t,\epsilon_b} := (k+R) q_{t,\epsilon_b}$ false elementary operation estimates in average. (7)

If there are $\leq \lceil e_{t,\epsilon_b} \rceil$ false operation guesses a brute force attack costs at most

$$w_{k+R,t,\epsilon_b} := \sum_{i=0}^{\lceil e_{t,\epsilon_b} \rceil} \binom{k+R}{i} \quad \text{trials (worst case estimate)} \tag{8}$$

to find and correct them. Table 3 supports the choice of an appropriate parameter t if the majority decision rule shall be applied in Phase 2. For $(k+R = 272, \epsilon_b = 0.05)$, for instance, 3-birthdays are sufficient while for $(k+R = 1040, \epsilon_b = 0.20)$ the parameter $t = 17$ seems appropriate. (Of course, smaller t is possible at cost of higher correction workload $w_{k+R,t,p}$; note further that $> \lceil e_{t,p} \rceil$ errors may

Table 3. Majority decision in Phase 2: Expected number of false operation estimates and guessing workload due to (8)

$k+R$	ϵ_b	t	q_{t,ϵ_b}	e_{t,ϵ_b}	$\log_2(w_{k+R,t,\epsilon_b})$	$k+R$	ϵ_b	t	q_{t,ϵ_b}	e_{t,ϵ_b}	$\log_2(w_{k+R,t,\epsilon_b})$
272	0.05	3	0.007	2.0	15.2	1040	0.05	5	0.001	1.2	19.1
272	0.10	5	0.008	2.3	21.7	1040	0.10	7	0.003	2.8	27.5
272	0.15	7	0.012	3.3	27.8	1040	0.15	11	0.003	2.8	27.5
272	0.20	11	0.012	3.2	27.8	1040	0.20	17	0.003	2.7	27.5
272	0.22	13	0.012	3.3	27.8	1040	0.25	27	0.002	2.5	27.5
272	0.24	15	0.013	3.7	27.8	1040	0.30	45	0.002	2.5	27.5

Table 4. Empirical results (The last column refers to the correct blinding classes)

					average number of			
t	k	R	ϵ_b	success rate	exponentiations	blinding classes	correct blinding classes	different r-values
3	256	10	0.05	10/10	150	138	138	138
5	256	10	0.10	10/10	672	483	475	475
7	256	10	0.15	10/10	1435	758	748	748
13	256	10	0.22	10/10	5253	1419	1246	1004
15	256	10	0.23	9/10	7425	1919	1228	989
15	256	10	0.24	1/10	6154	2017	1064	905
17	1024	10	0.20	10/10	6890	1037	1022	1022
27	1024	10	0.25	10/10	13833	1044	1026	1024
37	1024	10	0.28	6/10	23682	2004	1971	1024
45	1024	10	0.30	0/10	10742	2047	1807	1024

occur.) Table 4 collects experimental results. In order to reduce the workload we used the small blinding parameter $R = 10$. An attack was counted successful iff the (first) winning class was correct. Table 4 shows that for $R = 10$ errors rates up to 23% (256-bit ECC) and 28% (1024-bit RSA) are tolerable. Depending on ϵ_b we used decision boundaries $\gamma \in [\mu_Y - 4\sigma_Y, \mu_Y - 3\sigma_Y]$.

Remark 2. Possible improvements
(i) Resuming Phase 1 if the winning class $\mathcal{C}_w$ is incorrect might increase the success rate and thus the tolerable error rate ϵ_b.
(iii) Phase 1: For large t it might be an option to apply the decision strategy "if $(\mathrm{ham}(\widetilde{v}_j \oplus \widetilde{v}) < \gamma)$" to, let's say, at most 3 elements of $\mathcal{C}_m$ to reduce the probability of an erroneous decision $\widetilde{v}_j \notin \mathcal{C}_m$.
(iii) Phase 2: Depending on the target implementation for large t more efficient strategies than the majority decision rule might exist (e.g. DPA techniques), reducing t, thereby increasing the success rate and the maximal tolerable ϵ_b.

2.3 Efficiency, Scalability, and Limits

The term $N := 2^{\alpha R}$ with $\alpha := 1+(\log(t!)+1-R\log(2))/(Rt\log(2)-1)$ provides a rough estimate for the average number of traces needed until the first t-birthday occurs if all decisions are correct. (This formula is a result of a fruitful discussion

with E. Schulte-Geers.) For moderate R the value α ranges from 0.70 to 0.75 for $t = 3$, while $\alpha \approx 1$ and $\alpha > 1$ for medium-sized, resp. for large t, the exact value depending on (t, R). If wrong decisions occur, either spoiling existing blinding classes with wrong elements or opening new blinding classes unnecessarily (which might occur for large ϵ_b, cf. Table 4), more power traces are needed. For large t after $\approx 0.29 \cdot 2^R$ traces $0.25 \cdot 2^R$ blinding classes exist so that even in absence of wrong decisions the overall number of decisions on whether an exponent belongs to a particular blinding class is roughly in the order of 2^{2R}. Thus the number of traces and the number of decisions are limiting factors for $R \geq 32$ with large t. Since more correct decisions are necessary until $\widetilde{v}_j$ has been assigned to the correct blinding class, then γ has to be lowered to guarantee comparable overall success rate. Coarse heuristic considerations suggest that for $R = 16$, compared to $R = 10$, the tolerable error rate should drop down by roughly $\approx 3\%$ for ECC and $\approx 2\%$ for RSA. In fact, the setup $(\epsilon_b, k, R) = (20\%, 256, 16)$ showed 100% empirical success rate, matching with this heuristics. For $R = 32$ clearly further percentage points are lost.

3 Enhanced Attack Variant

The basic attack aims at t-birthdays of exponent blinding factors $r_1, r_2, \ldots$ where the choice of t depends on ϵ_b. This costs clearly more than $\sqrt{2^R}$ exponentiations, for large t even more than 2^R. A designer thus might feel secure just by selecting some value R for which the maximum number of exponentiations within the life cycle of the device (e.g., RSA signatures on a smart card) is clearly below $\sqrt{2^R}$. In contrast to the basic attack the enhanced variant does not aim at collisions of blinding factors but on collisions of sums of blinding factors, which can be obtained from substantially fewer exponentiations than $\sqrt{2^R}$.

3.1 Enhanced Attack: Overview

We begin with a definition. As in Section 2 the letter N denotes the sample size, i.e. the number of observed exponentiations, and $r_1, \ldots, r_N \in \{0, 1, \ldots, 2^R - 1\}$ are the blinding factors.

Definition 1. *For $u > 1$ let $M_u := \{(j_1, \ldots, j_u) \mid 1 \leq j_1 \leq \cdots \leq j_u \leq N\}$. For each u-tuple $(j_1, \ldots, j_u) \in M_u$ we define the 'u-sum' $S_u(j_1, \ldots, j_u) := r_{j_1} + \cdots + r_{j_u}$. For $(j_1, \ldots, j_u), (i_1, \ldots, i_u) \in M_u$ we write $(j_1, \ldots, j_u) \sim (i_1, \ldots, i_u)$ iff $S_u(j_1, \ldots, j_u) = S_u(i_1, \ldots, i_u)$.*

Observation. $\sim$ defines an equivalence relation on M_u. For $0 \leq m \leq u(2^R - 1)$ the equivalence class E_m consists of all u-tuples $(j_1, \ldots, j_u) \in M_u$ with $S_u(j_1, \ldots, j_u) = m$. Next we summarize the particular steps of the enhanced attack variant. With regard to N and error rate ϵ_b one first selects the parameter u. The parameters $u = 2, 3, 4$ should be most relevant for applications.

- Step 1: Identify several pairs of u-tuples $(j_1, \ldots, j_u), (j'_1, \ldots, j'_u) \in M_u$ with $S_u(j_1, \ldots, j_u) = S_u(j'_1, \ldots, j'_u)$, i.e. with $(j_1, \ldots, j_u) \sim (j'_1, \ldots, j'_u)$. This yields a system of linear equations over Z in the blinding factors $r_1, \ldots, r_N$.

– Step 2: Solve the system of linear equations gained in Step 1. More precisely, find $(r_1^*, \ldots, r_N^*)$ such that $(r_1, \ldots, r_N) = (r_1^*, \ldots, r_N^*) + (c, \ldots, c)$ for some unknown integer c much smaller than 2^R.
– Step 3:
  - ECC case: Compute $\widetilde{v}_j - r_j^* y = d^* + e_j$ with $d^* = d + c$ for $j \leq N$. Task: 1.) Find d^*. 2.) Determine d by exhaustive search in c.
  - RSA case: Compute $\widetilde{v}_j - \widetilde{v}_i = (r_j^* - r_i^*)y + (e_j - e_i)$ for several pairs (j, i). Task: Find y $(=\phi(n)$, resp. $= \phi(p))$.

3.2 NAF Representations

This subsection collects relevant properties of NAF representations, which will be needed in Step 1 of the enhanced attack.

Definition 2. $\mathrm{Z_m} := \{0, 1, \ldots, \mathrm{m} - 1\}$, *and if* $z = \sum_{j=0}^{t} \beta_j 2^j$ *with* $\beta_j \in \{1, 0, -1\}$ *for all* j *then* $(\beta_t, \ldots, \beta_0)_{SD}$ *is called a* binary signed-digit representation *of* z. *A* signed *representation is said to be* non-adjacent *if consecutive non-zero coefficients are always separated by at least one zero. We briefly speak of* NAF representations *where 'NAF' abbreviates 'non-adjacent form'. The* Hamming weight $\mathrm{ham}(\cdots)$ *of a signed-digit representation is the number of non-zero digits. The term* $\Phi(\cdot)$ *denotes the cumulative distribution function of the standard normal distribution.*

Every integer z has a unique non-adjacent representation, noted as $\mathrm{NAF}(z)$. In particular, $\mathrm{NAF}(z)$ has minimal hamming weight under all binary signed-digit representations of z. For $z < 0$ we set $\mathrm{NAF}(z) = -\mathrm{NAF}(|z|)$. Due to the lack of space we omit the proof of Lemma 1. Example 1 illustrates that the variance of $\mathrm{ham}(\mathrm{NAF}(X))$ is surprisingly small, which will be crucial for our attack.

Lemma 1. *For any integers* z_1 *and* z_2 *we have*

$$\mathrm{ham}(\mathrm{NAF}(z_1 + z_2)) \leq \mathrm{ham}(\mathrm{NAF}(z_1)) + \mathrm{ham}(\mathrm{NAF}(z_2)), \tag{9}$$

and the right-hand side is $\leq$ *the sum of the Hamming weights for any binary signed-digit representations* z_1 *and* z_2.
(ii) Let X *be a random variable uniformly distributed on* Z_{2^n} *with* $n \geq 2$. *Then*

$$\mathrm{Prob}\,(\mathrm{ham}(\mathrm{NAF}(X)) = k) = 2^{-n} \cdot \begin{cases} 1 & \text{for } k = 0 \\ n & \text{for } k = 1 \\ 2^{k-2}\left(\binom{n+1-k}{k} + \binom{n+2-k}{k}\right) & \text{for } k \in \{2, \ldots, \lfloor \frac{n+1}{2} \rfloor\} \\ 2^{\frac{n}{2}-1} & \text{for } k = \frac{n}{2} + 1, \text{ if } n \text{ is even} \end{cases} \tag{10}$$

(iii) Unless n *is too small, the random variable* $Y := \mathrm{ham}(\mathrm{NAF}(X))$ *is approximately normally distributed with expectation* $\mathrm{E}(Y) \approx 0.333n$ *and variance* $\mathrm{Var}(Y) \approx 0.075n$.

Example 1. $n = 1040$ (corresponds to 1024-bit RSA with $R = 16$), $Y := \mathrm{ham}(\mathrm{NAF}(X))$, $\mathrm{E}(Y) \approx 0.334n = 347.4$ $\sigma_Y := \sqrt{\mathrm{Var}(Y)} \approx \sqrt{0.075n} = 8.8$.

3.3 Enhanced Attack: Step 1

In Step 1 we have to decide whether $(j_1, \ldots, j_u) \sim (i_1, \ldots, i_u)$, i.e. whether $S_u(j_1, \ldots, j_u) = S_u(i_1, \ldots, i_u)$. Therefore, we apply the decision rule

$$\begin{aligned} &\text{Decide for } (j_1, \ldots, j_u) \sim (i_1, \ldots, i_u) \text{ iff} \\ &\mathrm{ham}(\mathrm{NAF}(\widetilde{v}_{j_1} + \cdots + \widetilde{v}_{j_u} - (\widetilde{v}_{i_1} + \cdots + \widetilde{v}_{i_u}))) < b_0 \quad \text{for some suitable } b_0. \end{aligned} \tag{11}$$

Motivation for the decision strategy (11). Clearly, $(\widetilde{v}_{j_1} + \cdots + \widetilde{v}_{j_u}) - (\widetilde{v}_{i_1} + \cdots + \widetilde{v}_{i_u}) = ry + e_{j_1} + \cdots + e_{j_u} - e_{i_1} + \cdots - e_{i_u}$ with $r = 0$ iff $(j_1, \ldots, j_u) \sim (i_1, \ldots, i_u)$ and $r \in \mathrm{Z} \setminus \{0\}$ else. By Lemma 1(iii) for a uniformly distributed random variable X on $\mathrm{Z}_{2^{k+R}}$ the variance $\mathrm{Var}(\mathrm{ham}(\mathrm{NAF}(X))) \approx 0.075(k+R)$ is very small, and thus $\mathrm{E}(\mathrm{ham}(\mathrm{NAF}(X)))$ is a 'typical' value. Hence for randomly selected v_j and random e_j one may expect $\mathrm{ham}(\mathrm{NAF}(\widetilde{v}_{j_1} + \cdots + \widetilde{v}_{j_u} - (\widetilde{v}_{i_1} + \cdots + \widetilde{v}_{i_u}))) \approx \mathrm{E}(\mathrm{ham}(\mathrm{NAF}(X))) \approx 0.333(k+R)$ if $(j_1, \ldots, j_u) \not\sim (i_1, \ldots, i_u)$ although the absolute differences of u-sums are not uniformly distributed on $Z_{2^{k+R}}$. Numerical experiments confirm the intuition (cf. Table 5). On the other hand, if $(j_1, \ldots, j_u) \sim (i_1, \ldots, i_u)$ by Lemma 1(i)

$$\begin{aligned} &\mathrm{ham}(\mathrm{NAF}(\widetilde{v}_{j_1} + \cdots + \widetilde{v}_{j_u} - \widetilde{v}_{i_1} - \cdots - \widetilde{v}_{i_u})) \leq \\ &\mathrm{ham}(\mathrm{NAF}(e_{j_1})) + \cdots + \mathrm{ham}(\mathrm{NAF}(e_{j_u})) - \cdots - \mathrm{ham}(\mathrm{NAF}(e_{i_u})) \leq \\ &\text{total number of guessing errors for } v_{j_1}, \ldots, v_{j_u}, v_{i_1}, \ldots, v_{i_u}. \end{aligned} \tag{12}$$

If the error rate is sufficiently small for $(j_1, \ldots, j_u) \sim (i_1, \ldots, i_u)$ the term $\mathrm{ham}(\mathrm{NAF}(\widetilde{v}_{j_1} + \cdots + \widetilde{v}_{j_u} - \widetilde{v}_{i_1} - \cdots - \widetilde{v}_{i_u}))$ is significantly smaller than for $(j_1, \ldots, j_u) \not\sim (i_1, \ldots, i_u)$, justifying decision rule (11). Each decision for $(j_1, \ldots, j_u) \sim (i_1, \ldots, i_u)$ gives a linear equation

$$S_u(j_1, \ldots, j_u) - S_u(i_1, \ldots, i_u) = r_{j_1} + \cdots + r_{j_u} - r_{i_1} - \cdots - r_{i_u} = 0. \tag{13}$$

Number of linear equations in Step 1. We derive a coarse estimator for the average number of (not necessarily independent) linear equations that one gains from N exponentiations if all decisions are correct. Neglecting those pairs of u-tuples $\in M_u$, which have at least two identical components, we conclude

$$\begin{aligned} \mathrm{E}(\#\text{linear equations}) &\approx \binom{\binom{N}{u}}{2} \mathrm{Prob}(S_u(1, \ldots, u) = S_u(u+1, \ldots, 2u)) \qquad (14) \\ &\approx \frac{N^{2u}}{2u!u!} \sum_{t=0}^{(2^R-1)u} \mathrm{Prob}(S_u(1, \ldots, u) = t)^2 \quad \text{for } u \ll N \end{aligned}$$

since for $u \ll N$ for the great majority of pairs $(j_1, \ldots, j_u), (i_1, \ldots, i_u)$ the components $j_1, \ldots, j_u, i_1, \ldots, i_u$ are mutually distinct. Approximately, the right-hand sum in (14) (without pre-factor $N^{2u}/(2u!u!)$) equals

$$\frac{c(u)}{2^R} \qquad \text{with } c(2) \approx \frac{2}{3}, \;\; c(3) \approx \frac{11}{20}, \;\; \text{and } c(u) \approx \frac{\sqrt{3}}{\sqrt{\pi u}} \text{ for } u \geq 4\,. \tag{15}$$

For $R = 16$, for instance, $N = 116$ for $u = 2$, resp. $N = 28$ for $u = 3$, resp. $N = 16$ (coarse estimate since $4 \not\ll 16$) for $u = 4$, exponentiations provide $\approx 2N$ linear equations (equate (14) with $2N$). Numerical experiments agree with these estimates, and these $N's$ roughly give the necessary number of power traces needed for the overall attack (cf. Phase 2, Table 6). Similarly, for $R = 32$ we need $N = 4689$ ($N = 141$, $N = 79$) power traces for $u = 2$ ($u = 3$, $u = 4$) to obtain $\approx 2N$ linear equations. In particular, $2N$ equations cost roughly

$$\approx \frac{N^{2u}}{2u!u!} \approx \frac{\left(\frac{2^R \cdot 4u!u!}{c(u)}\right)^{\frac{2u}{2u-1}}}{2u!u!} \approx 10 \cdot (2^R)^{1+\frac{1}{2u-1}} \quad \text{comparisons for } u \leq 4. \tag{16}$$

Discussion on the appropriate decision parameter. When applying decision rule (11) to u-tuples $(j_1, \ldots, j_u), (i_1, \ldots, i_u) \in M_u$ the attacker might commit two types of errors: False positives (decision for '$\sim$' although '$\not\sim$' is true) and false negatives (decision for '$\not\sim$' although '$\sim$' is true). A false negative just 'loses' one linear equation (13) whereas a false positive yields a wrong equation, making the solution of the system of linear equations (cf. Subsec. 3.4) meaningless.

Unlike in the basic version a false positive does not only slow down the attack (affecting a single blinding class) but the whole attack fails. False positives thus should definitely be prevented although the fact that $\sim$ defines an equivalence relation on M_u might allow to detect (and to cancel) some false positives Moreover, for the great majority of mutual comparisons '$\not\sim$' is the correct decision. More precisely, (14) to (16) says that only for the fraction $c(u)/2^R$ of all comparisons '$\sim$' is true. Assume that the random variables W and W' describe the distribution of $\mathrm{ham}(\mathrm{NAF}(\widetilde{v}_{j_1} + \cdots + \widetilde{v}_{j_u} - \widetilde{v}_{i_1} - \cdots - \widetilde{v}_{i_u}))$ under the condition $(j_1, \ldots, j_u) \not\sim (i_1, \ldots, i_u)$, resp. under the condition $(j_1, \ldots, j_u) \sim (i_1, \ldots, i_u)$. The empirical studies below consider $R = 16$ where we selected $b_0 := E(W) - 6.5\sigma_W$ in (11).

If we assume that W is at least approximately normally distributed the probability for a false positive in a single decision is $< 10^{-9}$. In Table 5 μ_W and $\mu_{W'}$ denote the mean values of W and W', while '$\sim$' and '$\not\sim$' briefly stand for $(j_1, \ldots, j_u) \sim (i_1, \ldots, i_u)$ and $(j_1, \ldots, j_u) \not\sim (i_1, \ldots, i_u)$, respectively. Each quintuple $(\mu'_W, \mu_W, \sigma'_W, \sigma_W, b_0)$ in Table 5 was figured on basis of 40000 pseudorandomly generated pairs $(\widetilde{v}_{j_1}, \ldots, \widetilde{v}_{j_u}), (\widetilde{v}_{i_1}, \ldots, \widetilde{v}_{i_u})$, fulfilling '$\sim$', resp. '$\not\sim$'. Table 5 verifies that the empirical values μ_W and σ_W are indeed very close to the expectation and standard deviation when X is uniformly distributed on $Z_{2^{k+R}}$ (cf. Lemma 1(iii) and Example 1), confirming our heuristics discussed at the beginning of this subsection. Table 5 shows for several pairs $(k + R, u)$ what error rates ϵ_b Step 1 of the enhanced attack can tolerate. As Phase 1 of the basic attack also Step 1 of the enhanced variant is more efficient for larger key sizes since the ratio $\sigma_W/(\mu_W - \mu_{W'})$ decreases as $(k + R)$ increases. We applied decision rule (11) in many simulation experiments. As predicted false positives did not occur for this choice of b_0. For $(k + R, u, \epsilon_b) = (1040, 3, 0.07)$ false negatives essentially do not occur whereas for $(k + R, u, \epsilon_b) = (272, 4, 0.04)$ about 29%

Table 5. ham(NAF(·))-u-sum-distributions for several sets of parameters

				ham(NAF($\widetilde{v}_{j_1} + \cdots - \widetilde{v}_{i_u}$))							ham(NAF($\widetilde{v}_{j_1} + \cdots - \widetilde{v}_{i_u}$))				
				$\sim$		$\not\sim$					$\sim$		$\not\sim$		
R	u	k	ϵ_b	$\mu_{W'}$	$\sigma_{W'}$	μ_W	σ_W	b_0	k	ϵ_b	$\mu_{W'}$	$\sigma_{W'}$	μ_W	σ_W	b_0
16	2	256	0.06	49.2	5.4	90.5	4.5	61.2	1024	0.11	272.7	10.4	347.0	8.8	289.8
16	2	256	0.07	54.7	5.4	90.1	4.5	61.0	1024	0.12	283.8	10.2	346.8	8.8	289.7
16	3	256	0.08	59.6	5.4	90.8	4.5	61.6	1024	0.13	293.4	10.1	346.8	8.8	289.9
16	3	256	0.04	48.7	5.4	90.5	4.5	61.2	1024	0.07	262.8	10.4	346.9	8.8	289.6
16	3	256	0.05	56.6	5.4	90.7	4.5	61.2	1024	0.08	279.8	10.2	346.9	8.8	290.1
16	3	256	0.06	63.0	5.4	90.8	4.5	61.7	1024	0.09	293.5	10.0	347.0	8.8	290.0
16	4	256	0.04	58.5	5.4	90.9	4.5	61.5	1024	0.06	277.7	10.3	347.0	8.8	290.0

(=$100(1 - \Phi((61.5 - 58.5)/5.4))\% = 100 \cdot (\Phi(0.56))\%$) of the existing (correct) equations get lost due to false negatives, increasing the required number of power traces N somewhat (cf. (15)).

Remark 3. At cost of efficiency (more decisions and more power traces) Step 1 of the enhanced attack also applies to larger error rates than discussed in Table 5. From (15) one concludes that $N \approx 116\sqrt[3]{a}$ for $u = 2$, $N \approx 28\sqrt[5]{a}$ for $u = 3$, and $N \approx 16\sqrt[7]{a}$ for $u = 4$ provide $\approx 2aN$ linear equations ($a > 1$), thus tolerating a fraction of $(a-1)/a$ lost equations. Numerical Example: For $(k + R, u, \epsilon_b) = (1040, 2, 0.12)$ we expect $\approx 28\%$ false negatives but $N = 129$ in place of $N = 116$ should compensate this defect.

3.4 Enhanced Attack: Step 2

Basic idea. In Step 1 the attacker has gained a homogeneous system of q linear equations

$$B\boldsymbol{r}' = \mathbf{0} \quad \text{with } B \in \mathrm{Mat}(q, N; \mathrm{Z}), \boldsymbol{r}' \in \mathrm{Z}^{\mathrm{N}}, \mathbf{0} = (0, \ldots, 0)^{\mathrm{t}} \in \mathrm{Z}^{\mathrm{q}}. \qquad (17)$$

In the following we assume that all equations are correct. This is the case if all decisions for '$\sim$' were correct, or if all false positives could be eliminated due to consistency checks within the equivalence classes $\mathcal{E}_m$. Then $\boldsymbol{r} := (r_1, \ldots, r_N)^t$ clearly is a solution of (17). Moreover, $z(1, \ldots, 1)^t \in \mathrm{Z}^{\mathrm{N}}$ is also a solution of (17) for all $z \in \mathrm{Z}$. As it is extremely unlikely that $\boldsymbol{r}$ is an integer multiple of $(1, \ldots, 1)^t$ (which can easily be checked) the kernel $\ker B$, the solution of (17), is a proper superset of $\{z(1, \ldots, 1)^t \mid z \in \mathrm{Z}\}$. Since $\dim(\ker(B)) \geq 2$ would be pointless to consider (17) as a system of linear equations over the real numbers.

Preliminary on Step 2

Definition 3. *Let $A \neq \{0\}$ be a commutative ring that operates on an abelian group $M \neq \{0\}$ via $(a, m) \mapsto am \in M$. Assume that besides $(ab)m = a(bm)$ and $1m = m$ also $a(m + n) = am + an$ and $(a + b)m = am + bm$ for all $a, b \in A$ and $m, n \in M$. Then M is called an A-module. The module M is called free if*

M admits an A-basis, i.e. if there exists a subset $B \subseteq M$ such that any $m \in M$ can be expressed in a unique way as an A-linear combination of elements of B. If M has a finite basis $\{m_1, \ldots, m_n\}$ then M has dimension n.

Note that if A is a field then M is a vector space. Lemma 2 illuminates the structure of $\ker B$ as a Z-module. We point out that for linear equations over $\mathbb{R}$ its assertion would be trivial while it is not obvious over Z.

Remark 4. (i) Example: Z^{n} becomes a Z-module via $y(z_1, \ldots, z_n) := (yz_1, \ldots, yz_n)$. (ii) Modules may not admit bases (e.g., the Z-module Z_{m}), and a set of independent elements may not be extendable to a basis: Consider, for example, the Z-module Z^2. The vector $(2, 0) \in \mathrm{Z}^2$ cannot be extended to a basis, and $(2, 0), (1, 0)$, for instance, only spans the sub-module $2\mathrm{Z} \times \mathrm{Z}$.
(iii) The definition of $\dim(M)$ tacitly uses the fact that all bases of M have the same cardinality since A is commutative (cf. [10], Chap. XIII).

Lemma 2. *$\ker B$ is a free sub-module of Z^{N}. In particular, $\ker B$ admits a module basis $\boldsymbol{s}_{(1)} := (1, \ldots, 1)^t, \boldsymbol{s}_{(2)}, \ldots, \boldsymbol{s}_{(\dim(\ker B))}$.*

Proof. Z^{N} is a free module over Z. As $\ker B$ is a sub-module of Z^{N} and since Z is a principal entire ring, $\ker B$ is also free and admits a basis with $\dim(\ker B)$ elements ([10], Chap. III, Thm. 7.1). It is $(1, \ldots, 1)^t, (0, 1, 0, \ldots, 0)^t, \ldots (0, 0, \ldots, 0, 1)^t$ is a basis of Z^{N}, and Z^{N} is a direct sum of the sub-modules T_1 and T_2 where T_1 is generated by $(1, \ldots, 1)^t$ and T_2 by basis vectors $(0, 1, 0, \ldots, 0)^t, \ldots (0, 0, \ldots, 0, 1)^t$. The mapping $\phi\colon T_1 \times T_2 \to \mathrm{Z}^{\mathrm{N}}, (\mathrm{t}_1, \mathrm{t}_2) \mapsto \mathrm{t}_1 + \mathrm{t}_2$ defines a module isomorphism. Then $\ker(B \circ \phi) = \{(t_1, t_2) \in T_1 \times T_2 \mid B(t_1 + t_2) = B(t_2) = 0\} = T_1 \times \{t_2 \in T_2 \mid B(t_2) = 0\}$. The second summand is a sub-module of $T_2 \cong \mathrm{Z}^{\mathrm{N}-1}$, is thus free and admits a basis $\boldsymbol{s}'_{(2)}, \ldots$ ([10], Chap. III, Thm. 7.1). Then $\boldsymbol{s}_{(1)} := \phi((1, \ldots, 1)^t, 0), \boldsymbol{s}_{(2)} := \phi(0, \boldsymbol{s}'_{(2)}), \ldots$ is a basis of $\ker B$, which proves the lemma.

If the number q of linear equations is sufficiently large, $\dim(\ker B) = 2$ may be expected. Experimental results confirm this conjecture (cf. Table 6). Larger parameter u allows smaller N but on the negative side the admissible error rate ϵ_b in Step 1 decreases. In the following we assume $\dim(\ker B)) = 2$. Then

$$\boldsymbol{r} = \mu_1 \boldsymbol{s}_{(1)} + \mu_2 \boldsymbol{s}_{(2)} \quad \text{with unknown } \mu_1, \mu_2 \in \mathrm{Z}. \tag{18}$$

Table 6. Enhanced attack: Experimental results for Step 2

	$u = 2$		$u = 3$		$u = 4$	
N	116	128	28	32	16	20
$\dim(\ker B) = 2$	43/50	49/50	49/50	50/50	12/50	50/50

Finding the basis. Finding μ_1 and μ_2 might seem to be a $O(2^{2R})$ problem, but this is not the case. Clearly, $|\mu_2| \cdot \max\{|\boldsymbol{s}_{(2)i} - \boldsymbol{s}_{(2)j}| : 1 \le i, j \le N\} = \max\{|r_i - r_j| : 1 \le i, j \le N\} < 2^R$. If the blinding factors $r_1, \ldots, r_N$ have been generated randomly it is very likely that the second term is $\approx 2^R$, and thus it is very unlikely that there is an integer μ_1 with $\gcd(r_1 - \mu_1, \ldots, r_N - \mu_1) > 1$. Thus almost certainly $\mu_2 \in \{-1, 0, 1\}$.

The case $\mu_2 = 0$ is very unlikely and easily checked. If $\mu_2 \neq 0$ order the indices $1, \ldots, N$ twice, first according to the size of the components $\boldsymbol{s}_{(2)1}, \ldots, \boldsymbol{s}_{(2)N}$ and secondly according to $\widetilde{v}_1, \ldots, \widetilde{v}_N$. If $\mu_2 = 1$ is correct both ordered sets should be similar while for $\mu_2 = -1$ both sets should become more similar if one order is reversed. Once μ_2 has been determined for μ_1 only

$$ca := 2^R - (\max\{|\boldsymbol{s}_{(2)i} - \boldsymbol{s}_{(2)j}| : 1 \le i, j \le N\} + 1) \text{ candidates remain} \quad (19)$$

to be checked. We assign to μ_1^* the minimal integer for which

$$\boldsymbol{r}^* = \mu_1^* \boldsymbol{s}_{(1)} + \mu_2 \boldsymbol{s}_{(2)} \quad \text{has only non-negative components.} \quad (20)$$

We point out that finding a Z-basis of $\ker B$ is more difficult than finding a basis of a vector space. In a first step we select some $\boldsymbol{v}_{(2)} \in \ker B$ that is no integer multiple of $\boldsymbol{s}_{(1)}$, where $\boldsymbol{v}_{(2)}$ is obtained by following steps.

1. Let $B' = \binom{B}{11\ldots1}$. Since B' is made by adding $\boldsymbol{s}_{(1)}$ to B, $\dim(\ker B') = 1$.
2. Use the Gaussian elimination to B', to obtain a reduced row echelon form

$$B'' = \begin{pmatrix} 1\,0\cdots0 & v'_{(2)1} \\ \vdots\,\vdots\;\;\vdots\,\vdots & \vdots \\ 0\,0\cdots1 & v'_{(2)N-1} \\ 0\,0\cdots0 & 0 \\ \vdots\,\vdots\;\;\vdots\,\vdots & \vdots \\ 0\,0\cdots0 & 0 \end{pmatrix}.$$

3. $\boldsymbol{v}_{(2)}$ is obtained by multiplying some integer to $(-v'_{(2)1}, \ldots, -v'_{(2)N-1}, 1)^t$.

If the greatest common divisor of its components exceeds 1, we divide $\boldsymbol{v}_{(2)}$ by this number. Lemma 3(i) provides a simple criterion to check whether $\{\boldsymbol{s}_{(1)}, \boldsymbol{v}_{(2)}\}$ is already a basis of $\ker B$. If not, Lemma 3(ii) explains how to get one, i.e. how to determine $\boldsymbol{s}_{(2)} = \boldsymbol{v}_{(3)}$. Due to lack of space we omit its proof.

Lemma 3. *Let* $\dim(\ker B) = 2$. *Assume further that that* $v_{(2)}$ *is no integer multiple of* $\boldsymbol{s}_{(1)}$, *and that the gcd over the components of* $\boldsymbol{v}_{(2)}$ *is* 1.
(i) Assertion (*) and Assertion (**) are equivalent:
(*) $\{\boldsymbol{s}_{(1)}, \boldsymbol{v}_{(2)}\}$ *is basis of* $\ker B$.
(**) There is no integer $1 < m \le 2\max\{|\boldsymbol{v}_{(2)1}|, \ldots, |\boldsymbol{v}_{(2)N}|\}$ *with*

$$\boldsymbol{v}_{(2)1} \equiv \cdots \equiv \boldsymbol{v}_{(2)N} (\mathrm{mod}\, m) \quad (21)$$

(ii) Assume that $\{\boldsymbol{s}_{(1)}, \boldsymbol{v}_{(2)}\}$ *is not a basis of* $\ker B$ *and that* m^* *is the largest integer with property (21). Then* $\{\boldsymbol{s}_{(1)}, \boldsymbol{v}_{(3)}\} := ((m^* - \boldsymbol{v}_{(2)1})\boldsymbol{s}_{(1)} + \boldsymbol{v}_{(2)})/m^*\}$ *is basis of* $\ker B$.

3.5 Enhanced Attack: Step 3

The goal of Step 3 is to find the secret key d. Recall that r^* denotes solution (20). Step 3 differs between ECC and RSA.

The ECC Case. The attacker first computes

$$a_j := \widetilde{v}_j - r_j^* y = d + (\mu_1 - \mu_1^*)y + e_j = d^* + e_j \qquad \text{for } j = 1, \ldots, N. \quad (22)$$

He knows all a_j while d^* and $e_j = (e_{j,n+R-1}, \ldots, e_{j,0})_2$ are unknown.
Algorithm ECC (finds d^*):

1. Clearly, $e_{j;0} = 0$ (correct guess of elementary operation 0 for exponent v_j) iff $d^* \equiv a_j (\mathrm{mod}\, 2)$. This leads to the following decision rule:
   (Majority Decision) For $i = 0, 1$ let $S_i := \{j \leq N \mid a_j \equiv i (\mathrm{mod}\, 2)\}$. If $|S_0| \geq |S_1|$ the attacker assumes that $e_{j',0} = 0$ for $j' \in S_0$ and $e_{j',0} = 1$ for $j' \in S_1$. The case $|S_0| < |S_1|$ is treated accordingly.
   (Error Correction): $e_{j,0} = 1$ either means that the attacker has guessed $\widetilde{v}_{j,0} = 1$ although $v_{j,0} = 0$ is true (Case I) or that the attacker has guessed $\widetilde{v}_{j,0} = 0$ although $v_{j,0} = 1$ is true (Case II). The attacker clearly can distinguish between both cases since he knows his guess $\widetilde{v}_{j,0}$.
   For Case I he replaces $\widetilde{v}_j$ by $\widetilde{v}_j - 1$ and thus in (22) a_j changes to $a_j - 1$. (Note: Since we are not interested in the e_j's we simply keep the notation e_j.) Similarly, for Case II one gets the new equation (22) $d^* + e_j = a_j + 1$. (Note that after error correction $d^* \equiv a_j (\mathrm{mod}\, 2)$ for all $j \leq N$.)
2. (Induction step): Assume that the guessing errors $e_{j,i}$ have already been corrected for $0 \leq i \leq k-1$. Thus $d^* \equiv a_j (\mathrm{mod}\, 2^k)$ for $j \leq N$. Now we are going to correct the $e_{j,k}$.
   (Majority Decision) Since $e_j \equiv 0 (\mathrm{mod}\, 2^k)$ equation $d^* + e_j \equiv a_j (\mathrm{mod}\, 2^{k+1})$ implies $d_k^* + e_{j,k} \equiv a_{j,k} (\mathrm{mod}\, 2)$. As in Step 1 the attacker applies the majority decision to identify those equations with a guessing error for elementary operation k.
   (Error Correction) as in Step 1, with $\pm 2^k$ in place of $\pm 1 = \pm 2^0$.

After this algorithm has terminated $d^* = a_1 = \cdots = a_N$ if all majority decisions had been correct. Since $d = d^* - (\mu_1 - \mu_1^*)y$ one known pair (P, dP) for some point P should suffice to identify d after at most ca trials (cf. (19)).

Remark 5. For error rate $\epsilon_b = \alpha$ one expects $e_{j,0} = 1$ in $\approx \alpha N$ equations but $e_{j,0} = 0$ in $\approx (1-\alpha)N$ equations. For $\alpha \leq 0.12$, for instance, $e_{1,i} + \cdots + e_{N,i} < N/2$ with overwhelming probability unless N is extremely small, and false majority decisions are very unlikely. For $N = 20$, $\alpha \leq 0.12$, the error probability is $\approx 10^{-5}$.

The RSA Case. At first the attacker re-labels $r_1^*, \ldots, r_N^*$ (and, in the same way, $\widetilde{v}_j, e_j$ etc.) such that $r_1^*, \ldots, r_{N'}^*$ have alternating parity (odd, even, odd, ... or even, odd, even,...) with maximal possible $N' \leq N$. The exponentiations $N'+1, \ldots, N$ are neglected in the following. Instead, we concentrate on a system of $N' - 1$ linear equations

$$b_j := \widetilde{v}_{j+1} - \widetilde{v}_j = \underbrace{(r^*_{j+1} - r^*_j)}_{=r_{j+1}-r_j} y + e_{j+1} - e_j := \Delta_j y + e_{j+1} - e_j \text{ for } 1 \le j < N'. \tag{23}$$

The integers b_j and Δ_j are known while y, e_j, e_{j+1} are unknown. By construction, $\Delta_j \equiv 1 (\mathrm{mod}\, 2)$ for all $j \le N' - 1$. The algorithm below applies the same techniques as the ECC algorithm but it is more complicated in detail.

Algorithm RSA (finds y ($= \phi(n)$ or $= \phi(p)$)):

1. Analogously to the ECC case the equations (23) are considered $(\mathrm{mod}\, 2)$. This gives modular equations $b_j \equiv y + e_{j+1} - e_j (\mathrm{mod}\, 2)$.
   (Majority Decision) For $i = 0, 1$ let $S_i := \{j < N' \mid b_j \equiv i (\mathrm{mod}\, 2)\}$. If $|S_0| \ge |S_1|$ the attacker assumes $b_{j'} \equiv y (\mathrm{mod}\, 2)$ and thus $e_{j'+1,0} - e_{j',0} \equiv 0 (\mathrm{mod}\, 2)$ for all $j' \in S_0$, and thus $e_{j'+1,0} + e_{j',0} \equiv 1 (\mathrm{mod}\, 2)$ for $j' \in S_1$. In particular, he guesses $y_0 := b_j \pmod 2$. The case $|S_0| < |S_1|$ is treated accordingly.
   (Error Correction): Majority Decision gives a system of $N' - 1$ linear equations $e_{j'+1,0} + e_{j',0} \equiv c_{j'} (\mathrm{mod}\ 2)$ where $c_{j'} = 0$ if j' belongs to the larger set S_i and $c_{j'} = 1$ else. Since the homogeneous system has 1-dimensional kernel $\{(0, \ldots, 0), (1, \ldots, 1)\}$ the non-homogeneous system has two solutions with mutually flipped components. The attacker decides for the solution with smaller Hamming weight. (If both Hamming weights are identical the solution is selected, which comes first in lexicographical order.)
   As for ECC $e_{j,0} = 1$ either means that the attacker has guessed $\widetilde{v}_{j,0} = 1$ although $v_{j,0} = 0$ is true (Case I) or that the attacker has guessed $\widetilde{v}_{j,0} = 0$ although $v_{j,0} = 1$ is true (Case II). The attacker replaces $\widetilde{v}_j$ by $\widetilde{v}_j - 1$ for CASE I, and by $\widetilde{v}_j + 1$ for Case II. Then he recalculates the left-hand sides of (23). (After error correction $y \equiv b_j (\mathrm{mod}\, 2)$ and $e_{j,0} = 0$ for all $j \le N'$.)
2. (Induction step) Assume that we have already guessed $y_0, \ldots, y_{k-1}$, and that we have corrected the guessing errors for the k least significant elementary operations, i.e. $e_j \equiv 0 (\mathrm{mod}\, 2^k)$ for all $j \le N'$. The next task is to guess y_k and to correct all guessing errors in $e_{1,k}, \ldots, e_{N',k}$. From (23) we conclude

$$b_j \equiv \widetilde{v}_{j+1} - \widetilde{v}_j = \Delta_j y + e_{j+1} - e_j \equiv$$
$$1 \cdot y_k 2^k + (\Delta_j - 1) \sum_{i=0}^{k-1} y_i 2^i + e_{j+1,k} 2^k - e_{j,k} 2^k (\mathrm{mod}\, 2^{k+1}). \tag{24}$$

   Since Δ_j is known and $y_0, \ldots, y_{k-1}$ have already been guessed we obtain

$$b'_j \equiv 2^k (y_k + e_{j+1,k} - e_{j,k}) (\mathrm{mod}\, 2^{k+1}) \tag{25}$$

   with $b'_j := b_j - (\Delta_j - 1) \sum_{i=0}^{k-1} y_i 2^i (\mathrm{mod}\, 2^{k+1})$. The right-hand side implies $b'_j = b'_{j,k} 2^k$, which yields

$$y_k + e_{j+1,k} - e_{j,k} \equiv b'_{j,k} (\mathrm{mod}\, 2) \tag{26}$$

   Analogously to Step 1 one estimates y_k (Majority Decision) and corrects the $e_{j,k}$ by modification of the $\widetilde{v}_j$ and recalculation of (23) (Error correction). The correction factor clearly is $\pm 2^k$ in place of ± 1.

Table 7. Success rates in Step 3 (computed from 300 trials each)

algo	$n+R$	ϵ_b	N	success rate	algo	$n+R$	ϵ_b	N	success rate
ECC	272	0.12	20	100%	RSA	1040	0.12	70	95%
ECC	272	0.12	16	96%	RSA	1040	0.12	60	85%
ECC	272	0.08	16	100%	RSA	1040	0.08	45	95%
ECC	272	0.08	10	91%	RSA	1040	0.08	35	74%

If all majority decisions have been correct the algorithm delivers y, which immediately gives the factorization of $n = pq$. We note that unlike (22) equations (23) are 'disturbed' by two error terms e_{j+1} and e_j, and some exponentiations are neglected. Consequently, for identical error rates the RSA algorithm requires larger sample size N than the ECC algorithm. Experiments (Table 7) verify that both error correction algorithms work. Note that the success rate depends only on (N, ϵ_b) but not on parameter u from Step 1. Table 7 shows that for both ECC and RSA for large error rate ϵ_b (e.g., $\epsilon_b = 0.12$, demanding $u = 2$) Step 1 and Step 2 determine the minimum number of power traces, which is needed for the overall attack. For RSA with small error rates ϵ_b (allowing $u > 2$) Step 3 defines the minimum number of power traces. Step 3 virtually does not depend on R, and thus for large R the minimum number of traces depends on Step 1 and Step 2.

3.6 Efficiency, Scalability, and Limits

The empirical results from Subsections 3.3 to 3.5 indicate that for $R = 16$ the enhanced attack tolerates error rates of 13%. Since the number of traces remains moderate even for large R the number of comparisons (16) is the limiting factor. However, the enhanced attack scales much better than the basic attack, and thus e.g. $R = 32$ is definitely feasible. For $R = 64$ even $u = 4$ requires roughly $\approx 2^{76}$ comparisons (NAF computations) in absence of false negatives, which might be viewed impractical for side-channel attacks, in particular since the attacker does not know in advance whether the error rate ϵ_b is indeed sufficiently small. Moreover, for increasing R the boundary b_0 has to be decreased (more comparisons per gained equation); replacing $6.5\sigma_W$, e.g. by $8.0\sigma_W$ (for $R = 32, u \geq 3$) or $8.5\sigma_W$ (for $R = 32, u = 2$), decreases the tolerable error rate by ≈ 1.5 percentage points. For $R = 48$ the reduction is ≈ 3 percentage points.

4 Conclusion

We introduced two variants of a new generic attack on exponent blinding (RSA without CRT, RSA with CRT, ECC). Experiments with simulated data confirmed the theoretical results. For small blinding factor $R = 10$ the basic attack was successful for error rates of $\epsilon_b \leq 0.23$ (256-bit ECC), resp. for $\epsilon_b \leq 0.28$ (1024-bit RSA). The enhanced attack works for $\epsilon_b \leq 0.13$ (for $R = 16$) but requires by orders of magnitudes less power traces and scales much better for large R. Moreover, the enhanced variant applies new techniques, which are interesting

by themselves. Of course, for security implementations power traces should not provide any useable information at all. However, selecting $R = 64$, or maybe even better $R > 64$, should make both variants impractical anyway. Limiting the use of the key clearly below $2^{R/2}$ even prevents 2-birthdays (basic attack).

Acknowledgement. The authors would like to thank David Galindo for his valuable support and the anonymous referees for their comments, which helped to improve the editorial quality of the paper.

References

1. Acıiçmez, O., Schindler, W.: A Vulnerability in RSA Implementations due to Instruction Cache Analysis and Its Demonstration on OpenSSL. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 256–273. Springer, Heidelberg (2008)
2. Fouque, P., Kunz-Jacques, S., Martinet, G., Muller, F., Valette, F.: Power Attack on Small RSA Public Exponent. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 339–353. Springer, Heidelberg (2006)
3. Henecka, W., May, A., Meurer, A.: Correcting Errors in RSA Private Keys. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 351–369. Springer, Heidelberg (2010)
4. Coron, J.S.: Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
5. Courrège, J.C., Feix, B., Roussellet, M.: Simple Power Analysis on Exponentiation Revisited. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 65–79. Springer, Heidelberg (2010)
6. Itoh, K., Izu, T., Takenaka, M.: Address-bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 129–143. Springer, Heidelberg (2003)
7. Itoh, K., Yamamoto, D., Yajima, J., Ogata, W.: Collision-Based Power Attack for RSA with Small Public Exponent. IEICE Transactions on Information and Systems E92-D5, #5, 897–908 (2009)
8. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
9. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
10. Lang, S.: Algebra, 3rd edn. Addison-Wesley, Reading (1993)
11. Schindler, W.: A Combined Timing and Power Attack. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 263–279. Springer, Heidelberg (2002)
12. Yen, S., Lien, W., Moon, S., Ha, J.: Power Analysis by Exploiting Chosen Message and Internal Collisions - Vulnerability of Checking Mechanism for RSA-Decryption. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 183–195. Springer, Heidelberg (2005)