Practical Key Recovery for Discrete-Logarithm Based Authentication Schemes from Random Nonce Bits

Aurélie Bauer[1] and Damien Vergnaud[2]

[1] Agence Nationale de la Sécurité des Systèmes d'Information
51 Boulevard de la Tour-Maubourg - 75700 Paris 07 SP, France
aurelie.bauer@ssi.gouv.fr
[2] École normale supérieure – C.N.R.S. – I.N.R.I.A.
45, rue d'Ulm, F-75230 Paris CEDEX 05, France

Abstract. We propose statistical cryptanalysis of discrete-logarithm based authentication schemes such as Schnorr identification scheme or Girault-Poupard-Stern identification and signature schemes. We consider two scenarios where an adversary is given some information on the nonces used during the signature generation process or during some identification sessions. In the first scenario, we assume that some bits of the nonces are known exactly by the adversary, while no information is provided about the other bits. We show, for instance, that the GPS scheme with 128-bit security can be broken using only 710 signatures assuming that the adversary knows (on average) one bit per nonce. In the second scenario, we assume that all bits of the nonces are obtained from the correct ones by independent bit flipping with some small probability. A detailed heuristic analysis is provided, supported by extensive experiments.

Keywords: Schnorr identification, Girault-Poupard-Stern identification, Girault-Poupard-Stern signatures, statistical cryptanalysis

1 Introduction

Since the introduction of the ElGamal signature scheme [6], many works have been devoted to design digital signatures and identification schemes, based on the *discrete logarithm problem* in a finite cyclic group $\mathbb{G}$ of order q (*e.g.* [14]). After Schnorr's proof of knowledge for discrete logarithms in groups of known prime order [22] (that can be used as an interactive identification scheme or be converted into a digital signature scheme using the Fiat-Shamir paradigm [8]), many other signature schemes have been designed, including the standard DSA [17]. Another variant, proposed by Girault [9] and further analysed by Poupard and Stern [21, 10] – called GPS – allows to use groups of unknown order.

All these signature and identification schemes perform randomized authentication: they use a *nonce* (or *ephemeral key*) r in $\{0, 1, \ldots, q-1\}$ for each message (or identification session) and compute g^r with g some generator of $\mathbb{G}$. These random nonces can be generated using either a (true) random number generator

or a pseudo-random one. Obviously extreme care is required in sampling such nonces since a predictable (or reused) output of the (pseudo)random generator may lead to a total break of the scheme. As an example, Bellare, Goldwasser and Micciancio mounted [2] a polynomial time key-recovery attack against the DSA signature scheme when the random nonces are generated using a (truncated) linear congruential generator. In [5], Bleichenbacher found a bias in the original DSA pseudorandom number generator specification, that could reveal the signer's private key and this attack was made practical recently [16].

Using *side-channel attacks*, partial information can be obtained on the nonce r from the run of the algorithm that computes g^r. In [15], Kuwakado and Tanaka proposed a polynomial-time algorithm that recovers the signer's private key given only two signatures with nonces smaller than $\sqrt{q}$ or where half of the nonces' bits are known to the adversary. Nguyen and Shparlinski [18] presented a polynomial-time algorithm that recovers the secret key of the signer when a few consecutive bits of the nonces are known for several signatures. Their algorithm runs in polynomial time when approximately $\sqrt{\log q}$ bits are known for a number of signatures in $O(\log q)$. Note that in practice, side-channel attacks will not generally reveal consecutive bits of the nonces used. Moreover, if countermeasures are used, only noisy information on the secrets may leak through power consumption (or other side-channel attacks) and an adversary may obtain partial information on the nonces (but not with perfect certainty).

In [11], it was reported that memory persistence times can be increased with simple cooling techniques and that an attacker with physical access to a machine may be able to recover some random part of cryptographic key information. Motivated by this work, Heninger and Shacham presented in [13] a new method for recovering RSA private keys given a fraction of private data (see also [20]). Their method succeeds with good probability in quadratic time if a fraction of at least 0.27 of the private key bits are known with certainty. In [12], Henecka, May and Meurer addressed the situation where no RSA private key bits are known with certainty but where a candidate for each secret key bit is known to the adversary in such a way that most bits are correct but some of them (unknown to the adversary) are flipped (in a symmetric way). In [19], Paterson, Polychroniadou and Sibborn brought a coding-theoretic viewpoint to bear on this problem. In particular, they highlighted the fact that the papers [13] and [12] considered the problem of error-correcting noisy RSA private keys in the binary erasure channel and the binary symmetric channel (respectively). This coding-theoretic viewpoint enabled them to design a new algorithm in another channel model more relevant for the so-called "cold-boot attacks" from [11] and to derive bounds on the performance of the proposed algorithms.

Contributions of the paper. We propose attacks on discrete-logarithm based authentication schemes where an adversary has some information on the nonces used in the signature generation or in some identification sessions. This information may come from the use of a biased (pseudo)random generator, a side-channel attack or a cold-boot attack. Using the coding-theoretic viewpoint from [19], we consider the following two scenarios:

– *erasure correction scenario:* we assume that some bits of the nonces are known exactly by the adversary, while no information at all is known about the other bits. This is defined in terms of a parameter δ representing the fraction of erasures. In standard coding terminology, this corresponds to an erasure model for errors, and an erasure channel.
– *error correction scenario:* we assume that all bits of the nonces are obtained from the correct ones by independent bit flipping with probability defined by a parameter also denoted δ. In coding-theoretic terms, this corresponds to a (memoryless) binary symmetric channel with crossover probability δ.

Our attacks apply to discrete-logarithm based authentication schemes where the "authentication relation" holds over the integers (instead of modulo the group order q). In particular, they can be applied readily to the GPS identification and the GPS signature schemes. However, they can also be mounted against Schnorr identification protocol if the adversary is first allowed to interact with the prover in a "dishonest" way before trying to impersonate her. In this *active attack model* [7, 4], which is the standard *de facto* security notion, an adversary can simply choose small challenges in the identification session in order to obtain an authentication relation that holds over the integers (after a small brute force search of the size of the chosen challenge).

In the erasure correction scenario, we provide an algorithm that, given t signatures or identification sessions (with $t \geq 2$) and partial information on the corresponding nonces with a fraction of erasures $\delta \simeq \ln(2)/t$, recovers the corresponding secret key in (heuristic) quadratic time. The algorithm recovers the nonces bit-by-bit, starting from the least significant bit to the most significant one, by growing a search tree and pruning it to remove partial solutions which do not match the known key bits. The algorithm is similar to the one proposed in [13] but works for any $t \geq 2$ and requires a more complex analysis. We implemented it and performed extensive experiments; in particular, we attack a 128-bit security level instantiation of GPS signature scheme that uses 512-bit nonces [21, 10] with $\delta \simeq 1/512$ (*i.e.* with on average only one nonce bit known for each signature). Our analysis guarantees that the private key can be recovered given only 710 signatures and a very naive implementation actually gives the secret key in a few minutes. In the error correction scenario, we provide an algorithm that, given t signatures or identification sessions (with $t \geq 2$) and partial information on the corresponding nonces with a fraction of errors $\delta \simeq 1/2 - \sqrt{\ln(2)/2t}$, recovers the corresponding secret key in (heuristic) quadratic time. We implemented our algorithm and performed extensive experiments using it. The attack analysis is simpler and follows closely the one presented in [12].

2 Preliminaries

Schnorr Identification. Let $\mathbb{G} = \langle g \rangle$ be a group of (known) prime order q and P and V denote a prover and a verifier. By engaging in the protocol, P proves to V that she knows the discrete logarithm x of a public group element $y = g^x$. The protocol has three simple moves:

Commitment P selects a random $r \in \{0, 1, \ldots, q-1\}$ and sends $k = g^r$ to V.
Challenge V picks a random $c \in \{0, 1, \ldots, q-1\}$ and sends c to P.
Response P computes and sends $s = r + cx \bmod q$ to V.

Eventually, V checks that $g^s \cdot y^{-c} = k$ and recognizes that P knows x if the equality holds. Schnorr's scheme is one of the most important ingredients in the design of cryptographic protocols and proofs of knowledge. It readily gives rise to an identification scheme where P proves her knowledge of the discrete logarithm of her public key.

The strongest form of attack against an identification scheme is an *active attack*, where the adversary (that wants to impersonate P) interacts with P, posing as V, but not necessarily following V's protocol. Since active attacks are quite feasible in practice, this model has become the standard *de facto* security notion for identification scheme. In [4], Bellare and Palacio proved that Schnorr identification scheme is secure against active attacks assuming the *one-more discrete logarithm assumption* in $\mathbb{G}$ (see [3]).

GPS Identification and Signature scheme. The GPS schemes were proposed by Girault in [9] and further analysed by Poupard and Stern [21, 10]. The schemes are similar to Schnorr's but allow to use groups of unknown order. Given a group $\mathbb{G} = \langle g \rangle$ generated by g, of (possibly) unknown order and three parameters $R, S, C \in \mathbb{N}$, the protocol has three simple moves to prove the knowledge of $x \in \{0, \ldots, S\}$ such that $y = g^x$:

Commitment P selects a random $r \in \{0, 1, \ldots, R\}$ and sends $k = g^r$ to V.
Challenge V picks a random $c \in \{0, 1, \ldots, C\}$ and sends c to P.
Response P computes and sends $s = r + cx$ to V.

Eventually, V checks that $g^s = ky^c$ and recognizes that P knows x if the equality holds and $s \in \{0, 1, \ldots, R + CS\}$. GPS signature scheme is derived from the identification scheme using the Fiat-Shamir heuristic [8]: to sign a message $m \in \{0,1\}^*$, the signer selects a random $r \in \{0, 1, \ldots, R\}$, computes $k = g^r$, computes $c = \mathcal{H}(m, k)$ where $\mathcal{H} : \{0,1\}^* \longrightarrow \{0, 1, \ldots, C\}$ is a cryptographic hash function and outputs the pair (s, c) where $s = r + cx$ as the signature of the message m. The security of the signature scheme against existential forgeries under chosen message attacks was proven (when $\mathcal{H}$ is modelled as a random oracle) in [10] under the assumption that computing discrete logarithms in $\mathbb{G}$ with exponents in $\{0, 1, \ldots, S\}$ is hard. For a k-bit security level, the analysis requires to use $S \simeq 2^{2k}$, $C \simeq 2^k$ and $R \simeq 2^{4k}$.

Authentication relation. In the following, we call the authentication relation of a discrete-logarithm based authentication scheme, the relation used in the **Response** phase of the three-move identification scheme (or in the signature generation protocol). Our attacks apply to all discrete-logarithm based authentication scheme for which this authentication relation holds over the integers. This is obviously the case for GPS identification and signature schemes. However, our

attack also applies to Schnorr identification if one considers an adversary allowed to mount an active attack and that has access to partial information on the nonce bits used in the different identification sessions. Such an adversary posing as V can indeed pick small challenges c (*e.g.* always $c = 1$). Since we have $s = r + cx \bmod q$ in Schnorr identification scheme, we obtain $s = r + cx + \alpha q$, where $\alpha \in \mathbb{Z}_-$ and $|\alpha| \leq c$. By performing an exhaustive search on the small set of values for α (*e.g.* $\alpha \in \{-1, 0\}$ when $c = 1$), the adversary obtains an authentication relation that holds over the integers.

Note that for all these discrete-logarithm based schemes, the knowledge of the random values generated during the signature process/identification session provides precious information on the signer's secret key. In particular, if an adversary is given access to a valid signature/identification transcript $\sigma = (s, c)$ (possibly for some message m), and to the nonce r such that $g^s = g^r y^c$ then he can retrieve the value of the signer's secret key x in polynomial time. Therefore, in the following, we will focus on attacks that aim to recover a complete nonce from partial information on nonces used in several signatures/identification sessions.

3 Erasure correction scenario

In this section, we focus on the *erasure correction scenario* described in the introduction where part of the bits of the random value r generated during the signature process are revealed to the attacker. To be more precise, we denote the binary decomposition of r as $r_0 + r_1 2^1 + \cdots + r_{n-1} 2^{n-1}$ and we introduce a parameter δ, with $0 \leq \delta \leq 1$, to define the probability that, at a given position $i \in \{0, \ldots, n-1\}$, the bit r_i is known to the adversary (and probability $(1 - \delta)$ that bit r_i is unknown).

Assuming that t signatures have been processed, we determine a lower-bound on δ allowing the attacker to fully recover the secret key x in (heuristic) polynomial time. We focus on the particular case of the GPS signature scheme, but as explained previously any other signature or identification scheme using authentication relations defined over the integers can be attacked similarly. In the following we first focus on the simple case of two signatures, and then generalize the study for a higher number of signatures.

3.1 The Attack Knowing Two Signatures

We assume that the adversary is given access to two valid signatures $\sigma_1 = (s_1, c_1)$ and $\sigma_2 = (s_2, c_2)$, respectively related to the messages m_1 and m_2. According to the description of the GPS scheme, provided in Section 2, the following relations hold:

$$s_1 = r_1 + c_1 x \text{ and } s_2 = r_2 + c_2 x \tag{1}$$

where r_1 and r_2 are nonces generated during the signature process and $c_i = \mathcal{H}(m_i, g^{r_i})$ (for $i \in \{1, 2\}$). Eliminating the unknown x in (1), we get:

$$C = r_1 c_2 - r_2 c_1 \tag{2}$$

where $C = s_1c_2 - s_2c_1$. Our algorithm will construct all pairs $(\vartheta_1, \vartheta_2)$ that satisfy (2) and match the known partial information on (r_1, r_2) - in the sequel we denote as $\mathcal{L}$ the list of such elements - and eventually, select, among them, the one that verifies the relations (1).

General Idea. Following [13], our method to construct the set $\mathcal{L}$ consists in performing an exhaustive search on all pairs $(\vartheta_1, \vartheta_2)$ satisfying Equation (2), guessing each of their bits from the least significant one to the most significant, and detecting invalid candidates during the process. Given a pair $(\vartheta_1, \vartheta_2)$ for which the relation (2) holds, if we denote $\vartheta_\alpha^{(i)} = \vartheta_{\alpha,0} + 2\vartheta_{\alpha,1} + \cdots + \vartheta_{\alpha,i}2^i$ where $\vartheta_\alpha^{(i)} = \vartheta_\alpha \bmod 2^{i+1}$ and $\vartheta_{\alpha,j} \in \{0,1\}$ for $\alpha \in \{1,2\}$ and $j \in \{0,\ldots,i\}$, we get:

$$\begin{cases} C = \vartheta_{1,0}c_2 - \vartheta_{2,0}c_1 \mod 2 \\ C = (\vartheta_{1,0} + 2\vartheta_{1,1})c_2 - (\vartheta_{2,0} + 2\vartheta_{2,1})c_1 \mod 2^2 \\ \quad \vdots \\ C = (\sum_{i=0}^{n-1} \vartheta_{1,i}2^i)c_2 - (\sum_{i=0}^{n-1} \vartheta_{2,i}2^i)c_1 \mod 2^n \end{cases} \tag{3}$$

Thus one can verify, at each step of the bit generations of $\vartheta_{1,i}$ and $\vartheta_{2,i}$, for i from 0 to $(n-1)$, whether the equation modulo 2^{i+1} of System (3) holds for the pair. If not, the corresponding bit-values for $(\vartheta_{1,i}, \vartheta_{2,i})$ are not kept as valid ones in the sequel of the bit-generations of $(\vartheta_{1,j}, \vartheta_{2,j})$ for j going from $(i+1)$ to $(n-1)$. This technique allows to reduce the size of the list $\mathcal{L}$ containing all final potential candidates $(\vartheta_1, \vartheta_2)$. In the following, we show that, in fact, this method leads to a polynomial time algorithm.

Description of the Technique. Let us now explain how to construct the list $\mathcal{L}$. In the analysis that follows, we use the notation $\mathcal{L}^{(k)}$ to refer to the state of list $\mathcal{L}$ at step k of the algorithm. Such a list will be defined as containing elements $(\vartheta_1, \vartheta_2)$ that are reduced modulo 2^{k+1}.

The basic principle of the algorithm can be sum up that way: the list $\mathcal{L}^{(0)}$ is first initialized with all pairs of bits $(\vartheta_{1,0}, \vartheta_{2,0})$ that satisfy Equation 1 of System (3) modulo 2 and that coincide with the bit-values preliminary known by the adversary on r_1 and r_2. All pairs $(\vartheta_{1,0}, \vartheta_{2,0})$ in $\mathcal{L}^{(0)}$ are then lifted to construct values modulo 2^2. To do so, one has to generate all possible values for pairs of bits $(\vartheta_{1,1}, \vartheta_{2,1})$ (which again coincide with the bits known by the adversary on $r_{1,1}$ and $r_{2,1}$) and to construct $\vartheta_1^{(1)}$ and $\vartheta_2^{(1)}$ respectively as $\vartheta_{1,0}+2\vartheta_{1,1}$ and $\vartheta_{2,0}+2\vartheta_{2,1}$. From now the obtained values $(\vartheta_1^{(1)}, \vartheta_2^{(1)})$ are checked to determine whether they satisfy Equation 2 of System (3) modulo 2^2. In case of an invalid answer, the corresponding bit-values for $(\vartheta_{1,1}, \vartheta_{2,1})$ are evicted and all remaining valid pairs $(\vartheta_1^{(1)}, \vartheta_2^{(1)})$ are put in $\mathcal{L}^{(1)}$. The process then continues from a bit-position i to the following one, taking all pairs $(\vartheta_1^{(i)}, \vartheta_2^{(i)})$ belonging to $\mathcal{L}^{(i)}$ at step i, lifting them by generating all possible bit-pairs $(\vartheta_{1,i+1}, \vartheta_{2,i+1})$ that coincide with the known bits on $r_{1,i+1}$ and $r_{2,i+1}$ and by creating new values $\vartheta_1^{(i+1)} = \vartheta_1^{(i)} + 2^{i+1}\vartheta_{1,i+1}$ and

$\vartheta_2^{(i+1)} = \vartheta_2^{(i)} + 2^{i+1}\vartheta_{2,i+1}$, and checking whether Equation $(i+2)$ of System (3) is satisfied modulo 2^{i+2}. Again, invalid solutions are evicted and the remaining pairs constitute the set $\mathcal{L}^{(i+1)}$. The algorithm finally stops when i equals n. A description of the whole process is provided in Algorithm 1.

Algorithm 1 Generic Attack on Two Signatures

Require: c_1, c_2, C and δ : partial bit information on r_1 and r_2
Ensure: $\mathcal{L}$ a list of pairs $(\vartheta_1, \vartheta_2)$ possible candidates for (r_1, r_2)

$\mathcal{L}^{(0)} = \{\}$

—/* *Initialisation - Case* $i = 0$ */—
Generate all bit pairs for $(\vartheta_{1,0}, \vartheta_{2,0})$: $\mathcal{E} = (0,0), (1,0), (0,1), (1,1)$
for each element $(\vartheta_{1,0}, \vartheta_{2,0}) \in \mathcal{E}$ **do**
  if $(\vartheta_{1,0}, \vartheta_{2,0})$ coincides with knowledge of $(r_{1,0}, r_{2,0})$ **then**
    if $(\vartheta_{1,0}, \vartheta_{2,0})$ satisfies Equation (1) of System (3) modulo 2 **then**
      Add element $(\vartheta_{1,0}, \vartheta_{2,0})$ to $\mathcal{L}^{(0)}$

—/* *Main loop for* i *from 1 to* $n-1$ */—
for i from 1 **to** $n-1$ **do**
  $\mathcal{L}^{(i)} = \{\}$
  Generate all bit pairs for $(\vartheta_{1,i}, \vartheta_{2,i})$: $\mathcal{E} = (0,0), (0,1), (1,0), (1,1)$
  for each element $(\vartheta_{1,i}, \vartheta_{2,i}) \in \mathcal{E}$ **do**
    if $(\vartheta_{1,i}, \vartheta_{2,i})$ coincides with knowledge of $(r_{1,i}, r_{2,i})$ **then**
      for each element $(\vartheta_1^{(i-1)}, \vartheta_2^{(i-1)}) \in \mathcal{L}^{(i-1)}$ **do**
        Lift $\vartheta_1^{(i)} = \vartheta_1^{(i-1)} + 2^i\vartheta_{1,i}$ and Lift $\vartheta_2^{(i)} = \vartheta_2^{(i-1)} + 2^i\vartheta_{2,i}$
        if $(\vartheta_1^{(i)}, \vartheta_2^{(i)})$ satisfies Equation $(i+1)$ of System (3) modulo 2^{i+1} **then**
          Add $(\vartheta_1^{(i)}, \vartheta_2^{(i)})$ to $\mathcal{L}^{(i)}$
return $\mathcal{L}^{(n-1)}$

Complexity Analysis. Estimating the overall complexity of Algorithm 1 can be reduced to the cost of constructing the list $\mathcal{L}^{(n-1)}$. Since this list has been built recursively from the previous ones $\mathcal{L}^{(n-2)}, \mathcal{L}^{(n-3)}, \ldots, \mathcal{L}^{(0)}$, we have to evaluate the expected cardinal of all $\mathcal{L}^{(i)}$ for i going from 0 to $(n-1)$ (denoted $\mathcal{N}^{(i)}$).

Let us first count the number of elements belonging to $\mathcal{L}^{(0)}$. By definition, this list contains all bit pairs $(\vartheta_{1,0}, \vartheta_{2,0})$ coinciding with the (possible) knowledge of $(r_{1,0}, r_{2,0})$ and satisfying Equation (1) of System (3). In fact, the number of solutions to that equation strongly relies on the parity of both c_1 and c_2. Indeed, two even values would give four pairs of solutions (in the general case) opposed to only two for odd values c_1 and c_2. For this reason, we have to split the analysis that follows in two scenarios, depending on the 2-adic valuation[1] of both c_1 and c_2. In the sequel, we denote as ℓ_1 (resp. ℓ_2) the 2-adic valuation of c_1 (resp. c_2).

- ***First analysis when $\ell_1 = \ell_2$***

Before coming back to the evaluation of the cardinality of the list $\mathcal{L}^{(0)}$, let us first see how the relation $\ell_1 = \ell_2$ impacts the shape of the equations belonging to System (3). When ℓ_1 and ℓ_2 are equal, Equation (2) can be simplified by dividing both sides of the equality by 2^{ℓ_1}. Indeed as this can be done for c_1 and

[1] We remind that the 2-adic valuation of a number c denotes the largest power of 2 dividing c.

c_2, this is obviously also the case for C. Thus, the shape of the relation does not change and the new obtained constants $c_1/2^{\ell_1}$ and $c_2/2^{\ell_1}$ are odd. In the following, for the sake of simplicity (and as this does not change the analysis, at the cost of renaming the variables), we still work with Equation (2) (and thus with System (3)) but assuming that the constants c_1 and c_2 are odd.

From now on, one can easily describe the elements that belong to $\mathcal{L}^{(0)}$ as bit-values of the form $(\vartheta_{1,0}, \vartheta_{2,0})$ satisfying $C = \vartheta_{1,0} + \vartheta_{2,0} \mod 2$, and such that $(\vartheta_{1,0}, \vartheta_{2,0})$ coincide with possible knowledge of $(r_{1,0}, r_{2,0})$. This description allows to evaluate $\mathcal{N}^{(0)}$, see Lemma 1 (a proof can be found in the full version of the paper [1]).

Lemma 1. *We have* $\mathcal{N}^{(0)} = \delta^2 - 2\delta + 2$.

We now have to find, for a fixed i between 1 and $(n-1)$, an expression of $\mathcal{N}^{(i)}$ in function of $\mathcal{N}^{(i-1)}$. By definition $\mathcal{L}^{(i)}$ can be described as the set of elements $(\vartheta_1^{(i)}, \vartheta_2^{(i)})$ defined as $\vartheta_1^{(i)} = \vartheta_1^{(i-1)} + 2^i\vartheta_{1,i}$ and $\vartheta_2^{(i)} = \vartheta_2^{(i-1)} + 2^i\vartheta_{2,i}$ with $(\vartheta_1^{(i-1)}, \vartheta_2^{(i-1)}) \in \mathcal{L}^{(i-1)}$, $(\vartheta_{1,i}, \vartheta_{2,i}) \in \{0,1\}^2$ and satisfying Equation $(i+1)$ of System (3), namely:

$$C = (\vartheta_1^{(i-1)} + 2^i\vartheta_{1,i})c_2 - (\vartheta_2^{(i-1)} + 2^i\vartheta_{2,i})c_1 \mod 2^{i+1} \tag{4}$$

Moreover, the pairs $(\vartheta_{1,i}, \vartheta_{2,i})$ should coincide with possible information on $(r_{1,i}, r_{2,i})$. Knowing that $(\vartheta_1^{(i-1)}, \vartheta_2^{(i-1)})$ belong to $\mathcal{L}^{(i-1)}$, the relation $C = \vartheta_1^{(i-1)}c_2 - \vartheta_2^{(i-1)}c_1 \mod 2^i$ necessarily holds. As a consequence, there exists an integer $k \in \mathbb{Z}$ such that $C = \vartheta_1^{(i-1)}c_2 - \vartheta_2^{(i-1)}c_1 + k2^i$. Putting this relation into Equation (4) and simplifying the whole expression allows to reach the following new condition:

$$k = \vartheta_{1,i} + \vartheta_{2,i} \mod 2 \tag{5}$$

The choice of $(\vartheta_{1,i}, \vartheta_{2,i})$ solutions to (5) and coinciding with possible information on $(r_{1,i}, r_{2,i})$, strongly depends on whether $(\vartheta_1^{(i-1)}, \vartheta_2^{(i-1)})$ equals $(r_1^{(i-1)}, r_2^{(i-1)})$ or not. Indeed, if these values are equal, namely $(\vartheta_1^{(i-1)}, \vartheta_2^{(i-1)})$ is the beginning of the *right* solution, then when $r_{1,i}$ and $r_{2,i}$ are known, Equation (5) necessarily holds (obviously as this is the searched solution). In that case, one will choose $(\vartheta_{1,i}, \vartheta_{2,i}) = (r_{1,i}, r_{2,i})$. To the contrary, when $(\vartheta_1^{(i-1)}, \vartheta_2^{(i-1)}) \neq (r_1^{(i-1)}, r_2^{(i-1)})$, a value is fixed for k and the knowledge of $(r_{1,i}, r_{2,i})$ does not necessarily make Equation (5) be satisfied. In that case, the choice $(r_{1,i}, r_{2,i})$ will not be maintained. As we cannot determine whether Equation (5) would be satisfied or not, we introduce a new parameter $\gamma \in [0,1]$, which is only defined when $(\vartheta_1^{(i-1)}, \vartheta_2^{(i-1)}) \neq (r_1^{(i-1)}, r_2^{(i-1)})$, corresponding to the probability that Equation (5) holds. A detailed discussion on the value of γ is given in Section 3.3 (see also the full version of the paper [1]).

We can finally express the number of elements belonging to $\mathcal{L}^{(i)}$ in function of those of $\mathcal{L}^{(i-1)}$, as claimed by the following lemma.

Lemma 2. *Under our heuristic, we have* $\mathcal{N}^{(i)} = \mathcal{N}^{(i-1)}(\gamma\delta^2 - 2\delta + 2) + \delta^2(1-\gamma)$.

A proof of this lemma can be found in the full version of the paper, see [1]. Combining Lemmas 1 and 2, we obtain:

$$\begin{aligned}\mathcal{N}^{(n-1)} &= \mathcal{N}^{(n-2)}(\gamma\delta^2 - 2\delta + 2) + \delta^2(1-\gamma)\\ &\vdots\\ &= (\gamma\delta^2 - 2\delta + 2)^{n-1}(\delta^2 - 2\delta + 2) + \tfrac{\delta^2(1-\gamma)}{2\delta - \gamma\delta^2 - 1}(1 - (\gamma\delta^2 - 2\delta + 2)^{n-1})\end{aligned}$$

The goal of the adversary being to construct the set $\mathcal{L}^{(n-1)}$ in polynomial time, this attack will only be made practical if the quantity $(\gamma\delta^2 - 2\delta + 2)$ is strictly smaller than 1. In that case, the cardinality of the list $\mathcal{L}^{(n-1)}$ will not grow too fast when n tends toward infinity. Since γ is unknown, we have to evaluate it in order to reach some necessary condition on δ. Setting γ to $1/2$ seems to be a reasonable choice, see Section 3.3. We finally reach the condition $\frac{1}{2}\delta^2 - 2\delta + 2 < 1$, which is satisfied for $\delta > 2 - \sqrt{2} \simeq 0.59$. By taking this value as a lower bound on δ, we are able to determine the expected size of the set $\mathcal{L}^{(n-1)}$, namely $(3 - 2\sqrt{2})n + 1$. Knowing that n refers to the bit-size of the random values generated during the signature process, we thus obtain a polynomial time complexity for our attack.

Theorem 1 (Two Signatures and $\ell_1 = \ell_2$). *An adversary able to learn a proportion of $\delta = (2 - \sqrt{2})$ bits of the random nonces used during the generation of two known signatures can (heuristically) break the scheme in polynomial time. In that case, the expected space required for performing the attack is* $(3 - 2\sqrt{2})n^2 + 1 \simeq 0.17n^2 + 1$.

- ***Second analysis when $\ell_1 \neq \ell_2$.***

The study is a bit more tedious here, but the analysis can be adapted to prove Theorem 2. The entire proof is provided in the full version of the paper [1].

Theorem 2 (Two Signatures and $\ell_1 < \ell_2$). *An adversary, able to learn a proportion of $\delta = (2 - \sqrt{2})$ bits of the random nonces used during the generation of two known signatures, can (heuristically) break the scheme in polynomial time. In that case, the expected space required for performing the attack is* $(\sqrt{2} - 1)n^2 + 1 \simeq 0.41n^2 + 1$.

When comparing Theorem 1 and Theorem 2, one notices that the lower bound on δ is the same for both cases "$\ell_1 = \ell_2$" and "$\ell_1 < \ell_2$". However the space required to construct the sets $\mathcal{L}^{(i)}$ is higher when $\ell_1 \neq \ell_2$ than in the other case. This actually impacts the efficiency of the attack since the number of constructed solutions is larger.

3.2 The Attack Knowing t Signatures

Let us now consider the case of an adversary that is given access to t signatures $\sigma_1, \ldots, \sigma_t$ corresponding to known messages $m_1, \ldots, m_t$. We denote $\sigma_i = (s_i, c_i)$

the signatures and r_i the nonces used in their generation ($s.t.\ c_i = \mathcal{H}(m_i, g^{r_i})$) for i in $\{1, \ldots, t\}$. We denote $r_{i,j}$ the j-th bit of r_i for each j in $\{1, \ldots, n\}$ and i in $\{1, \ldots, t\}$. We have $s_i = r_i + c_i \cdot x$ for $i \in \{1, \ldots, t\}$ and we obtain:

$$\begin{cases} C_{1,2} = r_1 c_2 - r_2 c_1 \\ \quad\vdots \\ C_{1,t} = r_1 c_t - r_t c_1 \end{cases} \tag{6}$$

where $C_{1,j} = s_1 c_j - s_j c_1$ for $j \in \{2, \ldots, t\}$. As above, our algorithm retrieves the nonces $r_1, \ldots, r_t$ by collecting all tuples $(\vartheta_1, \ldots, \vartheta_t)$ satisfying System (6) and coinciding with possible knowledge on the bits $r_{i,j}$. As previously, $\mathcal{L}$ denotes the set containing such elements[2]. The complexity of the attack can thus be reduced to the cost of constructing $\mathcal{L}$.

General Idea. As for the "two-signature case", a way to retrieve $(r_1, \ldots, r_t)$ would consist in performing an exhaustive search on all bit-values $r_{i,j}$, for i going from 1 to t and j from 0 to $(n-1)$, using some possible additional information on the bits $r_{i,j}$, and to detect the invalid candidates during the process. The technique consists in generating all tuples $(\vartheta_1^{(k)}, \ldots, \vartheta_t^{(k)})$ satisfying System (6) modulo 2^{k+1} and to select among them, the one that can be lifted to solutions modulo 2^{k+2}. Of course, these operations should be consistent with the possible knowledge on some bits of the r_i's. In this algorithm, parameter k will vary from 0 to $(n-1)$. At the end of the procedure, the set $\mathcal{L}^{(n-1)}$ containing all elements $(\vartheta_1^{(n-1)}, \ldots, \vartheta_t^{(n-1)})$ satisfying System (6) modulo 2^n and coinciding with known information on the bits $r_{i,j}$, is in fact the desired one: $\mathcal{L}$. A precise description of the whole method is provided in Algorithm 2.

Complexity Analysis. Let us now determine the number of elements belonging to $\mathcal{L}$ by evaluating the expected cardinal of all $\mathcal{L}^{(i)}$ for i going from 0 to $(n-1)$ (denoted $\mathcal{N}^{(i)}$). We start with $\mathcal{L}^{(0)}$ which, by definition, contains all tuples $(\vartheta_{1,0}, \ldots, \vartheta_{t,0})$ in $\{0,1\}^t$ coinciding with possible knowledge on bits $(r_{1,0} \ldots, r_{t,0})$ and verifying System (6) modulo 2. One notices that the number of such solutions strongly depends on the 2-adic valuations of $c_1, \ldots, c_t$. For this reason, and similarly to the "two signatures case", we split the analysis in two configurations depending on whether the 2-adic valuations of $c_1, \ldots, c_t$ are all equal or not. In the rest of the paper, we denote as ℓ_i the 2-adic valuation of c_i (for $i \in \{1, \ldots, t\}$).

- ***First analysis with $\ell_1 = \ell_2 = \cdots = \ell_t$***

When the 2-adic valuations are all equal, System (6) can be simplified by dividing each equation by 2^{ℓ_1}. One thus reaches a new system involving simpler equations,

[2] Obviously once $(r_1, \ldots, r_t)$ has been retrieved, the signers' secret key x can easily be recovered. Theoretically only one of the r_i's is really necessary to retrieve x, but we see in the following that recovering one such element requires in fact to retrieve all r_is simultaneously.

Algorithm 2 Generic Attack on t Signatures

Require: $(c_1, \ldots, c_t)$, $(C_{1,1}, \ldots, C_{1,t})$, δ and partial bit information on $(r_1, \ldots, r_t)$
Ensure: $\mathcal{L}$ a list of tuples $(\vartheta_1, \ldots, \vartheta_t)$ possible candidates for $(r_1, \ldots, r_t)$
1: $\mathcal{L}^{(0)} = \{\}$
2:
—/* *Initialisation - Case* $k = 0$ */—
3: Generate all bit tuples for $(\vartheta_{1,0}, \ldots, \vartheta_{t,0})$, say $\mathcal{E} = \{0,1\}^t$
4: **for** each element $(\vartheta_{1,0}, \ldots, \vartheta_{t,0}) \in \mathcal{E}$ **do**
5: **if** $(\vartheta_{1,0}, \ldots, \vartheta_{t,0})$ coincides with knowledge of $(r_{1,0}, \ldots, r_{t,0})$ **then**
6: **if** $(\vartheta_{1,0}, \ldots, \vartheta_{t,0})$ satisfies System (6) modulo 2 **then**
7: Add element $(\vartheta_{1,0}, \ldots, \vartheta_{t,0})$ to $\mathcal{L}^{(0)}$
8:
—/* *Main loop for* k *from 1 to* $n-1$ */—
9: **for** k from 1 **to** $n-1$ **do**
10: $\mathcal{L}^{(k)} = \{\}$
11: Generate all bit tuples for $(\vartheta_{1,k}, \ldots, \vartheta_{t,k})$, say $\mathcal{E} = \{0,1\}^t$
12: **for** each element $(\vartheta_{1,k}, \ldots, \vartheta_{t,k}) \in \mathcal{E}$ **do**
13: **if** $(\vartheta_{1,k}, \ldots, \vartheta_{t,k})$ coincides with knowledge of $(r_{1,k}, \ldots, r_{t,k})$ **then**
14: **for** each element $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)}) \in \mathcal{L}^{(k-1)}$ **do**
15: Lift $\vartheta_1^{(k)} = \vartheta_1^{(k-1)} + 2^k \vartheta_{1,k}$
$\vdots$
16: $\vdots$
17: Lift $\vartheta_t^{(k)} = \vartheta_t^{(k-1)} + 2^k \vartheta_{t,k}$
18: **if** $(\vartheta_1^{(k)}, \ldots, \vartheta_t^{(k)})$ satisfies System (6) modulo 2^{k+1} **then**
19: Add $(\vartheta_1^{(k)}, \ldots, \vartheta_t^{(k)})$ to $\mathcal{L}^{(k)}$
20: **return** $\mathcal{L}^{(n-1)}$

say $C'_{1,j} = r_1 d_j - r_j d_1$ where $C'_{1,j} = C_{1,j}/2^{\ell_1}$ is known to the attacker. To simplify the analysis that follows - and since it does not change anything but renaming the variables - we assume that we keep working with System (6) but using odd values c_i. Now if we come back to our analysis on the set $\mathcal{L}^{(0)}$, namely considering System (6) modulo 2, we reach the following new system:

$$\begin{cases} C_{1,2} = \vartheta_{1,0} + \vartheta_{2,0} \mod 2 \\ \vdots \qquad\qquad\qquad \vdots \\ C_{1,t} = \vartheta_{1,0} + \vartheta_{t,0} \mod 2 \end{cases} \tag{7}$$

It now becomes easy to count the number of elements belonging to $\mathcal{L}^{(0)}$. Indeed, either none of the $r_{i,0}$ is known by the adversary, and in that case there are two possible values for $\vartheta_{1,0}$, say 0 or 1 and each of them fixes the rest for $\vartheta_{2,0}, \ldots, \vartheta_{t,0}$. In the second configuration, when there is at least one of the $r_{i,0}$ which is known by the attacker (say $r_{1,0}$ for instance[3]), the value $\vartheta_{1,0}$ is fixed to $r_{1,0}$ and thus the other values $\vartheta_{2,0}, \ldots, \vartheta_{t,0}$ are fixed too (see System (7)). As a consequence, there is unique solution for the uple $(\vartheta_{1,0}, \ldots, \vartheta_{t,0})$. The construction of the whole set $\mathcal{L}^{(0)}$ is summed up in Algorithm 2a, which can be seen as an adaptation of the initialisation phase of Algorithm 2 when all ℓ_i are equal (process to a replacement of lines 3–7 of Algorithm 2 by Algorithm 2a).
The expected cardinal $\mathcal{N}^{(0)}$ of the list $\mathcal{L}^{(0)}$ can thus be expressed easily:

– when none of the $r_{i,0}$'s is known, which holds with probability $(1-\delta)^t$, there are two candidates for $(\vartheta_{1,0}, \ldots, \vartheta_{1,t})$.

[3] One can easily reorder the random values r_i to make such an assumption true.

Algorithm 2a Init. phase of Algorithm 2 when $\ell_1 = \cdots = \ell_t$ (lines 3–7)

if (None of the $r_{i,0}$'s is known) **then**
  Add $\{(0, C_{1,2} \mod 2, \ldots, C_{1,t} \mod 2), (1, C_{1,2}+1 \mod 2, \ldots, C_{1,t}+1 \mod 2)\}$ to $\mathcal{L}^{(0)}$
else
  /*At least one of the $r_{i,0}$'s is known, say for instance $r_{1,0}$*/
  Add $(r_{1,0}, C_{1,2}+r_{1,0} \mod 2, \ldots, C_{1,t}+r_{1,0} \mod 2)$ to $\mathcal{L}^{(0)}$

– when at least one of the $r_{i,0}$ is known, what happens with probability $1-(1-\delta)^t$, there is a unique solution for $(\vartheta_{1,0}, \ldots, r_{t,0})$.

One can thus reach $\mathcal{N}^{(0)} = 2(1-\delta)^t + 1 - (1-\delta)^t = 1 + (1-\delta)^t$. We will now determine by induction the expected size $\mathcal{N}^{(k)}$ from $\mathcal{N}^{(k-1)}$ for $k \in \{1, \ldots, n-1\}$, knowing that it contains all elements of the form $(\vartheta_1^{(k)}, \ldots, \vartheta_t^{(k)})$ coinciding with possible knowledge on $(r_{1,k}, \ldots, r_{t,k})$ and satisfying:

$$\begin{cases} C_{1,2} = (\vartheta_{1,0} + \cdots + \vartheta_{1,k}2^k) \cdot c_2 + (\vartheta_{2,0} + \cdots + \vartheta_{2,k}2^k) \cdot c_1 \mod 2^{k+1} \\ \quad \vdots \qquad\qquad\qquad\qquad\qquad \vdots \\ C_{1,t} = (\vartheta_{1,0} + \cdots + \vartheta_{1,k}2^k) \cdot c_t + (\vartheta_{t,0} + \cdots + \vartheta_{t,k}2^k) \cdot c_1 \mod 2^{k+1} \end{cases} \tag{8}$$

Putting such expressions inside System (8) finally leads to the following new relations:

$$k_j = \vartheta_{1,k} + \vartheta_{j,k} \mod 2 \text{ for } j \in \{1, \ldots, t\} \tag{9}$$

Now it is easier to determine the number of solutions $(\vartheta_{1,k}, \ldots, \vartheta_{t,k})$ that will be chosen to lift elements $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)})$ from $\mathcal{L}^{(k-1)}$ to $\mathcal{L}^{(k)}$. Nevertheless one should be careful during this analysis, since it strongly depends on whether the chosen element $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)})$ equals $(r_1^{(k-1)}, \ldots, r_t^{(k-1)})$ or not. Indeed when this condition is not satisfied, the choice of an element $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)})$ fixes all integers $k_1, \ldots, k_t$ implying some possible *invalid* restrictions on the values $\vartheta_{1,k}, \ldots, \vartheta_{t,k}$. As a consequence, when some of the $r_{i,k}$'s are known (precisely more than two), the corresponding equalities in (9) are not necessarily satisfied. Since we do not know in advance when it happens, we assume each equation in (9) holds *independently* with some fixed probability $\gamma \in [0,1]$ during the whole run of our algorithm[4]. In the other case, namely when the $r_{i,k}$'s are unknown or when the element $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)})$ corresponds to the *right* solution, the lifting process behaves as usual, see the "two signatures case". Taking all these considerations into account, one can finally determine the size of the set $\mathcal{L}^{(k)}$, see Algorithm 2b for a precise description (this algorithm can be seen as an adaptation of the main loop of Algorithm 2, lines 10 –19).
From now on, we are able to deduce the size $\mathcal{N}^{(k)}$ of the set $\mathcal{L}^{(k)}$, knowing that:

– when the adversary does not know any of the $r_{i,k}$, what holds with probability $(1-\delta)^t$, there are $2\mathcal{N}^{(k-1)}$ solutions;
– when exactly one of the $r_{i,k}$ is known, which happens with probability $t\delta(1-\delta)^{t-1}$, there are $\mathcal{N}^{(k-1)}$ solutions;

[4] We remind that γ is only defined when $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)}) \neq (r_1^{(k-1)}, \ldots, r_t^{(k-1)})$

– in the other case, this number depends on the element $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)})$. When this element is equal to $(r_1^{(k-1)}, \ldots, r_t^{(k-1)})$ (this is the *right candidate*), there is a unique solution. In the other case, there are $\gamma(\mathcal{N}^{(k-1)} - 1)$ solutions when exactly two $r_{i,k}$ are known, what holds with probability $\binom{t}{2}\delta^2(1-\delta)^{(t-2)}$; there are $\gamma^2(\mathcal{N}^{(k-1)} - 1)$ solutions when exactly three $r_{i,k}$ are known, which holds with probability $\binom{t}{3}\delta^3(1-\delta)^{(t-3)}$; and so on; there are $\gamma^{t-1}(\mathcal{N}^{(k-1)} - 1)$ solutions when all the $r_{i,k}$ are known, what happens with probability $\binom{t}{t}\delta^t$.

Algorithm 2b Main loop of Algorithm 2 when $\ell_1 = \cdots = \ell_t$ (lines 10 – 19)

$\mathcal{L}^{(k)} = \{\}$
if none of $r_{1,k}, \ldots, r_{t,k}$ is known **then**
  for each element $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)}) \in \mathcal{L}^{(k-1)}$ **do**
    for each element $\vartheta_{1,k}$ in $\mathcal{E} = \{0, 1\}$ **do**
      Construct $\vartheta_1^{(k)} = \vartheta_1^{(k-1)} + 2 \cdot \vartheta_{1,k}$
      for each index j in $\{2, \ldots, t\}$ **do**
        Compute k_j such that $k_j \cdot 2^k = C_{1,j} - \vartheta_1^{(k-1)} 2^{k-1} c_j - \vartheta_j^{(k-1)} 2^{k-1} c_1$
        Compute $\vartheta_{j,k} = \vartheta_{1,k} + k_j \mod 2$; construct $\vartheta_j^{(k)} = \vartheta_j^{(k-1)} + 2\vartheta_{j,k} \mod 2^{k+1}$
      Add element $(\vartheta_1^{(k)}, \ldots, \vartheta_t^{(k)})$ in $\mathcal{L}^{(k)}$
else if there exists a unique i such that $r_{i,k}$ is known /**say for instance* $r_{1,k}$*/ **then**
  for each element $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)}) \in \mathcal{L}^{(k-1)}$ **do**
    Construct $\vartheta_1^{(k)} = \vartheta_1^{(k-1)} + 2 \cdot \vartheta_{1,k}$
    for each index j in $\{2, \ldots, t\}$ **do**
      Compute k_j such that $k_j \cdot 2^k = C_{1,j} - \vartheta_1^{(k-1)} 2^{k-1} c_j - \vartheta_j^{(k-1)} 2^{k-1} c_1$
      Compute $\vartheta_{j,k} = \vartheta_{1,k} + k_j \mod 2$; construct $\vartheta_j^{(k)} = \vartheta_j^{(k-1)} + 2\vartheta_{j,k} \mod 2^{k+1}$
    Add element $(\vartheta_1^{(k)}, \ldots, \vartheta_t^{(k)})$ in $\mathcal{L}^{(k)}$
else
  /**two or more* $r_{i,k}$ *are known, assume* $r_{1,k}$ *is concerned**/
  for each element $(\vartheta_1^{(k-1)}, \ldots, \vartheta_t^{(k-1)}) \in \mathcal{L}^{(k-1)}$ **do**
    for each index j in $\{1, \ldots, t\}$ **do**
      Compute k_j such that $k_j \cdot 2^k = C_{1,j} - \vartheta_1^{(k-1)} 2^{k-1} c_j - \vartheta_j^{(k-1)} 2^{k-1} c_1$
    Set $\vartheta_{1,k} = r_{1,k}$
    for each index j in $\{2, \ldots, t\}$ **do**
      Compute $\vartheta_{j,k} = \vartheta_{1,k} + k_j \mod 2$
    if $(\vartheta_{1,k}, \ldots, \vartheta_{t,k})$ coincide with possible knowledge on $(r_{1,k}, \ldots, r_{t,k})$ **then**
      for each index j in $\{1, \ldots, t\}$ **do**
        Compute $\vartheta_j^{(k)} = \vartheta_j^{(k-1)} + 2 \cdot \vartheta_{j,k} \mod 2^{k+1}$
      Add element $(\vartheta_1^{(k)}, \ldots, \vartheta_t^{(k)})$ to $\mathcal{L}^{(k)}$

By combining all these results, we obtain the following formula for $\mathcal{N}^{(k)}$:

$$2(1-\delta)^t \mathcal{N}^{(k-1)} + (1 - (1-\delta)^t) + \frac{1}{\gamma}(\mathcal{N}^{(k-1)} - 1) \sum_{i=1,\ldots,t} \binom{t}{i} (\gamma\delta)^i (1-\delta)^{t-i}$$

Evaluating the quantity inside the summation leads to the following relation:

$$\mathcal{N}^{(k)} = \mathcal{N}^{(k-1)} \left(\frac{1}{\gamma}(\gamma\delta + 1 - \delta)^t + (2 - \frac{1}{\gamma})(1-\delta)^t \right) + 1 - \frac{1}{\gamma}(\gamma\delta + 1 - \delta)^t + \left(\frac{1}{\gamma} - 1 \right) (1-\delta)^t$$

Denoting as $\mathcal{A}(\gamma,\delta)$ the quantity $\frac{1}{\gamma}(\gamma\delta+1-\delta)^t+(2-\frac{1}{\gamma})(1-\delta)^t$ and using the formula obtained for $\mathcal{N}^{(0)}$, we reach:

$$\begin{aligned}\mathcal{N}^{(n-1)} &= (1+(1-\delta)^t)\mathcal{A}(\gamma,\delta)^{n-1} \\ &\quad + \left(1-\tfrac{1}{\gamma}(\gamma\delta+1-\delta)^t+(\tfrac{1}{\gamma}-1)(1-\delta)^t\right)\textstyle\sum_{i=0\ldots n-2}\mathcal{A}(\gamma,\delta)^i\end{aligned}$$

The goal of the adversary being to construct the set $\mathcal{L}^{(n-1)}$ in polynomial time, this attack will only be made practical if the quantity $\mathcal{A}(\gamma,\delta)$ is strictly smaller than 1. In that case, the size of the list $\mathcal{L}^{(n-1)}$ will not grow too fast when n tends toward infinity. Setting γ to $1/2$ (see Section 3.3 for experimental results on that point) leads to the condition $2(1-\frac{\delta}{2})^t<1$, which is satisfied for $\delta>2-2^{1-1/t}$. By taking this value as a lower bound on δ, we are able to determine the expected maximum size of the set $\mathcal{L}^{(n-1)}$, namely $(2^{1-\frac{1}{t}}-1)^t n+1$.

Theorem 3 (t signatures and $\ell_1=\cdots=\ell_t$). *An adversary able to learn a proportion of $\delta=2-2^{1-1/t}$ bits of the random nonces used in the generation of t known signatures can (heuristically) break the scheme in polynomial time. In that case, the expected space required for performing the attack is $(2^{1-\frac{1}{t}}-1)^t n^2+1$.*

- ***Second analysis when some of the ℓ_i's are different.***

The analysis is more tedious here but can still be adapted to the case when some of the $\ell_{\mathbf{i}}$'s are different, see Theorem 4 below (the proof is provided in the full version of the paper [1]).

Theorem 4 (t signatures with ℓ_i different). *An adversary able to learn a proportion of $\delta=2-2^{1-1/t}$ bits of the random nonces used in the generation of t known signatures can (heuristically) break the scheme in polynomial time. In that case, the expected space required for performing the attack[5] is $(2^{1-\frac{1}{t}}-1)^e n^2+1$.*

Remark 1. Using the coding viewpoint from [19], one can obtain limits on the performance of any algorithm for selecting candidate nonces in the erasure correction scenario. Their argument is based on the converse to Shannon's noisy-channel coding theorem [6]. The underlying code is made of the 2^n words on tn bits obtain by the naive algorithm without pruning (*i.e.* with code rate $1/t$) so simple variants of our algorithm cannot be efficient for $\delta<1/t$ and our algorithm is optimal up to the multiplicative constant $\ln(2)$.

3.3 Experimental Results

To confirm the validity of the attack and of our heuristic, extensive experiments have been performed for various values of t and δ. For each pair (t,δ), the attack

[5] Here, the index e is defined such that $\ell_1=\ell_2=\cdots=\ell_e<\ell_{e+1}\leq\cdots\leq\ell_t$.

[6] This theorem states that no combination of code and decoding procedure can jointly achieve arbitrarily reliable decoding when the code rate exceeds the capacity of the channel.

has been launched and its complexity has been measured by counting the sum of the cardinality of the sets $\mathcal{L}^{(i)}$ for i in $\{0, \ldots, n\}$. The obtained results are analysed below.

With 32 and 64 signatures. We performed experiments using 32 and 64 signatures (using a security parameter κ equal to 128). Since the experiments were time-consuming, we performed 1000 experiments for each pair (t, δ). The results for $t = 32$ are provided on Figure 1, for values of δ varying from 0.04 to 0.06 and the ones for $t = 64$ are illustrated on Figure 2, for values of δ between 0.02 and 0.04 (namely from 2% to 4%). These experiments show that it works better in practice than what was predicted. Indeed, the bounds below which the attack begins to become unpractical are approximately 0.04 for $t = 32$ signatures and 0.02 for $t = 64$. These values are better than the ones reached by the theoretical analysis, say $2 - 2^{1-1/32} \simeq 0.043$ and $2 - 2^{1-1/64} \simeq 0.022$.

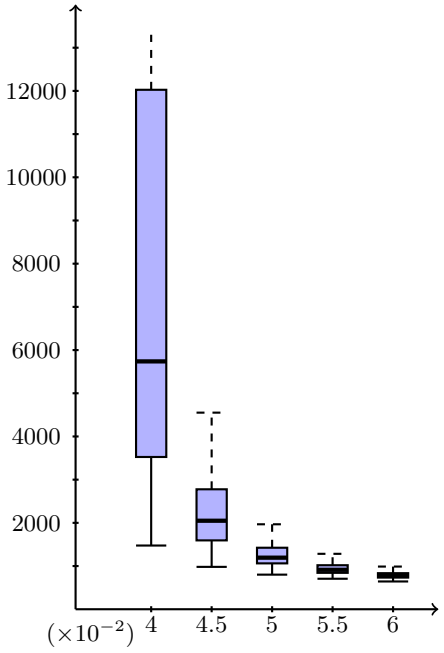

Fig. 1. Total number of constructed $(\vartheta_1, \ldots, \vartheta_{32})$ in function of δ ($t = 32$ and $\kappa = 128$).

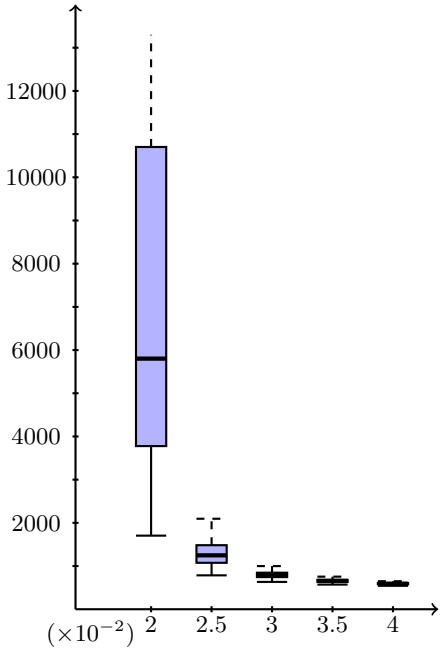

Fig. 2. Total number of constructed $(\vartheta_1, \ldots, \vartheta_{64})$ in function of δ ($t = 64$ and $\kappa = 128$).

With 710 signatures. We finally considered the case where the adversary knows (on average) only one bit per nonce r_i (*i.e.* $\delta = 1/512$). For a security parameter κ equals to 128, our theoretical analysis claims that an attacker will

be able to break the scheme knowing 710 signatures. In this case, our attack succeeded in a few minutes (using a very naive implementation) with a total size of generated elements equal to 2437. To go further, we also tried to launch the attack for a smaller number of signatures, but still keeping $\delta = 1/512$. The cases $t = 650$ and $t = 600$ were also successful. The time required for the last case was approximately a few hours and the total sum of elements was 53770 (again using a very naive implementation).

Impact of γ. The formula obtained in Section 3.1 and 3.2 strongly depend on the value of the parameter γ. In our theoretical analysis, we decided to take $\gamma = 1/2$. In practice, several tests have been launched to determine whether this assumption is correct. In every experiments we performed, the observed value for γ was very near from $1/2$, what confirms our initial assumption.

4 Error correction scenario

In the *erasure correction scenario* (see previous sections), we compared each nonce candidate with its given fragmentary version in order to determine the nonce uniquely with overwhelming probability. In this section, the *error correction scenario*, we proceed similarly: we assume an adversary obtain some error-prone nonces in a discrete-logarithm based authentication scheme which are derived from the actual nonces by flipping each bit individually with some fixed probability $\delta \in [0, 1/2)$. Intuitively, if δ is below $1/2$, then among all nonce candidates, the Hamming distance between the least significant bits of the actual nonces and the nonces with noise should be minimal. The analysis is simpler than in the previous section and relies on the technique proposed by Henecka, May and Meurer in [12].

We consider the case of an adversary that is given access to t signatures (or authentication sessions) $\sigma_1, \ldots, \sigma_t$ corresponding to known messages $m_1, \ldots, m_t$. Again, each signature σ_i is defined as a pair (s_i, c_i) where $s_i = r_i + c_i x$ and x is the signers' secret key. Moreover we assume that each bit of each nonce r_i is known with some error: to formalize this model we denote by δ the probability that for each nonce r_i, its j-th bit $r_{i,j}$ is flipped (for j in $\{1, \ldots, n\}$). By definition, the bit $r_{i,j}$ is correct with probability $(1 - \delta)$.

Making the quantity x disappear from all the expressions, one reaches the simple system of equations (6). The adversary can retrieve the nonces $r_1, \ldots, r_t$ by collecting all tuples $(\vartheta_1, \ldots, \vartheta_t)$ satisfying System (6) and minimizing the Hamming distance with the bits $r_{i,j}$. The algorithm performs a clever exhaustive search, consisting in generating all tuples $(\vartheta_1^{(k)}, \ldots, \vartheta_t^{(k)})$ satisfying System (6) modulo 2^{k+1} and then selecting among them, those that can be lifted to solutions modulo 2^{k+2}. These operations should minimize the distance with the knowledge on some bits of the r_i's.

Contrary to the erasure correction scenario, one cannot easily prune partial solutions that do not coincide with the known secret key bits. Indeed this process may discard the correct solution, since this solution does not fully agree with

the noisy nonce material. Thus, in this scenario, we do no longer compare bit by bit but we compare larger blocks of bits. More precisely, we grow subtrees of depth T (for some parameter T) for each t-tuple of nonce candidate. We will see that this results in 2^T new candidates which we all compare with our noisy nonce material. If the Hamming distance with this material in these T bits is above some threshold parameter d we keep the candidate, otherwise we discard it. The only difficulty here consists in estimating these parameters T and d:

- the parameter T cannot be chosen too large since in each iteration the search tree grows by a factor 2^T ;
- the parameter T has to be sufficiently large in order to guarantee that the actual t-tuple of nonces has a small Hamming distance with the noisy nonce material (in each block of successive T bits) but incorrect t-tuples of nonces are separable by the threshold parameter d ;
- the threshold parameter d has to be large enough to guarantee that with probability close to 1 the actual t-tuple of nonces will never be discarded during the execution of the algorithm ;
- the parameter d cannot be chosen too large since otherwise we obtain too many faulty t-tuples candidates for the next iteration.

- ***First analysis with $\ell_1 = \ell_2 = \cdots = \ell_t$.***

As above, one can assume we are using odd values c_i. We have to determine the size $\mathcal{N}^{(k)}$ of the sets $\mathcal{L}^{(k)}$. For $k = 0$, since none of the $r_{i,0}$ is known for sure by the adversary, there are two possible values for the bit $\vartheta_{1,0}$ and each of them fixes the other bits $\vartheta_{2,0}, \ldots, \vartheta_{t,0}$ (using System (7)). Therefore, we obtain $\mathcal{N}^{(0)} = 2$. Similarly, using System (9), one can see that if there is no pruning at the depth k then $\mathcal{N}^{(k)}$ is simply equal to $2\mathcal{N}^{(k-1)}$.

- ***Second analysis when some of the ℓ_i's are different.***

In this case, we can still show that $\mathcal{N}^{(0)} = 2$ and that we have, before pruning[7], $\mathcal{N}^{(T-1)} = 2^T$ at iteration T (see [1] for a complete proof of this result).

We now consider the pruning phase. Let us define a random variable X_c for the number of matching bits between the actual t-tuple of nonces and the noisy nonce material in a block of T consecutive bits. Clearly X_c is the binomial distribution with parameters tT and probability $(1-\delta)$: $\Pr[X_c = m] = \binom{tT}{m}(1-\delta)^m \delta^{tT-m}$ for m in $\{0, \ldots, tT\}$.

Considering an incorrect partial solution for the t-tuple of nonces, we denote X_b for the number of matching bits between the expansion of this incorrect solution by T bits and the noisy nonce material in the corresponding block of T consecutive bits. We make use of the following heuristic assumption.

Heuristic 1 *Every solution generated by applying the expansion phase to an incorrect partial solution is an ensemble of T randomly chosen bit slices.*

[7] In particular, in both cases, one gets before pruning $\mathcal{N}^{(T-1)} = 2^T$ at iteration T (instead of $\mathcal{N}^{(T-1)} = 2^{tT}$ for a naive approach).

Therefore every expansion of an incorrect candidate in the expansion phase results in tT uniformly random bits. We verified the validity of this heuristic experimentally. Under this assumption, we obtain $\Pr[X_b = m] = \binom{tT}{m}(2)^{-tT}$ for $m \in \{0, \ldots, tT\}$.

Henecka, May and Meurer proved in [12, Main Theorem 7] that these conditions are sufficient to insure the existence of an expansion parameter T and a threshold parameter d such that the two distributions are sufficiently separated and the growing factor 2^T in the expansion phase is polynomial.

Theorem 5 ([12]). *Under the previous heuristic, for every $\epsilon > 0$, the following holds: let $n, t \geq 2$ be two integers, let $T = \left\lceil \frac{\ln(n)}{t\epsilon^2} \right\rceil$, $\gamma = \sqrt{(1 + 1/T)\frac{\ln(2)}{2t}}$, and $d = tT\left(\frac{1}{2} + \gamma\right)$. An adversary able to learn (in the error correction scenario) the individual bits of random nonces of t known signatures of length n with probability $\delta < \frac{1}{2} - \gamma - \epsilon$ can recover the nonces (and therefore the secret key) in expected time $O(n^{2+\ln(2)/t\epsilon^2})$ with success probability at least $1 - (m\epsilon^2/\ln(n) + 1/n)$.*

Remark 2. The following table gives the limit crossover probability δ (*i.e.* with $\epsilon \longrightarrow 0$ and $T \longrightarrow +\infty$) in the error correction scenario depending on the number of signatures/identification sessions available t known to the adversary:

t	2	3	4	5	6	7	8	9	t
δ	0.084	0.160	0.205	0.237	0.260	0.277	0.292	0.303	$1/2 - \sqrt{\ln(2)/2t}$
δ^*	0.110	0.174	0.214	0.243	0.264	0.281	0.295	0.306	$H_2^{-1}(1 - 1/t)$

The value δ^* corresponds to the optimal value of δ one can derive from the converse to Shannon's noisy-channel coding theorem as in Remark 1 using the fact that the channel capacity of (memoryless) binary symmetric channel with crossover probability δ is $1 - H_2(\delta)$ where $H_2(\delta) = -\delta \log_2(\delta) - (1-\delta)\log_2(1-\delta)$ is the entropy function. For a 128-bit security level, with nonces of binary length 512, each signature provides only one bit of information for $\delta \simeq 1/2 - 10^{-4}$ and one needs in theory around $70 \cdot 10^6$ signatures in order to recover the secret key.

5 Conclusion

In this paper, we proposed attacks on discrete-logarithm based authentication schemes where an adversary has some information on the nonces used during the signature generation process or during some identification sessions, in the *erasure correction scenario* and the *error correction scenario*. The following table sums up the limit crossover probability δ in the two scenarios depending on the number N_r of signatures/identification sessions known to the adversary:

N_r	2	3	4	5	6	7	t
δ (erasure correction)	0.586	0.413	0.318	0.259	0.218	0.188	$\simeq \ln(2)/t$
δ (error correction)	0.084	0.160	0.205	0.237	0.260	0.277	$1/2 - \sqrt{\ln(2)/2t}$

Our methods can be generalized to the Z-channel considered in [19].

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

References

1. Aurélie Bauer and Damien Vergnaud. Practical key recovery for discrete-logarithm based authentication schemes from random nonce bits. Full version of the paper, Cryptology ePrint Archive, 2015.
2. Mihir Bellare, Shafi Goldwasser, and Daniele Micciancio. "pseudo-random" number generation within cryptographic algorithms: The DDS case. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 277–291. Springer, August 1997.
3. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.
4. Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer, August 2002.
5. D. Bleichenbacher. On the generation of one-time keys in dl signature schemes. Presentation at IEEE P1363 Working Group meeting, November 2000. Unpublished.
6. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, August 1984.
7. Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.
8. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, August 1986.
9. Marc Girault. Self-certified public keys. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer, April 1991.
10. Marc Girault, Guillaume Poupard, and Jacques Stern. On the fly authentication and signature schemes based on groups of unknown order. *Journal of Cryptology*, 19(4):463–487, October 2006.
11. J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 45–60. USENIX Association, 2008.
12. Wilko Henecka, Alexander May, and Alexander Meurer. Correcting errors in RSA private keys. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 351–369. Springer, August 2010.
13. Nadia Heninger and Hovav Shacham. Reconstructing RSA private keys from random key bits. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 1–17. Springer, August 2009.
14. Patrick Horster, Holger Petersen, and Markus Michels. Meta-El-Gamal signature schemes. In *ACM CCS 94: 2nd Conference on Computer and Communications Security*, pages 96–107. ACM Press, 1994.

15. Hidenori Kuwakado and Hatsukazu Tanaka. On the security of the elgamal-type signature scheme with small parameters. *IEICE Transactions*, 82-A(1):93–97, 1999.
16. Elke De Mulder, Michael Hutter, Mark E. Marson, and Peter Pearson. Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 435–452. Springer, August 2013.
17. National Institute of Standards and Technology. *FIPS PUB 186-2: Digital Signature Standard (DSS)*. National Institute for Standards and Technology, Gaithersburg, MD, USA, January 2000.
18. Phong Q. Nguyen and Igor Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15(3):151–176, 2002.
19. Kenneth G. Paterson, Antigoni Polychroniadou, and Dale L. Sibborn. A coding-theoretic approach to recovering noisy RSA keys. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 386–403. Springer, December 2012.
20. Colin Percival. Cache missing for fun and profit. In *Proceedings of BSDCan 2005*.
21. Guillaume Poupard and Jacques Stern. On the fly signatures based on factoring. In *ACM CCS 99: 6th Conference on Computer and Communications Security*, pages 37–45. ACM Press, November 1999.
22. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.