Extended Hidden Number Problem and Its Cryptanalytic Applications

Martin Hlaváč[1,⋆] and Tomáš Rosa[1,2]

[1] Department of Algebra, Charles University in Prague,
Sokolovská 83, 186 75 Prague 8, Czech Republic
[2] eBanka, a.s., Václavské Náměstí 43, 110 00 Prague 1, Czech Republic
hlavm1am@artax.karlin.mff.cuni.cz, trosa@ebanka.cz

Abstract. Since its formulation in 1996, the Hidden Number Problem (HNP) plays an important role in both cryptography and cryptanalysis. It has a strong connection with proving security of Diffie-Hellman and related schemes as well as breaking certain implementations of DSA-like signature schemes. We formulate an extended version of HNP (EHNP) and present a polynomial time algorithm for solving its instances. Our extension improves usability of HNP for solving real cryptanalytic problems significantly. The techniques elaborated here can be used for cryptographic strength proving, as well. We then present a practically feasible side channel attack on certain implementations of DSA (e.g. OpenSSL), which emphasizes the security risk caused by a side channel hidden in the design of Pentium 4 HTT processor for applications like SSH. During experimental simulations, having observed as few as 6 authentications to the server, an attacker was able to disclose the server's private key.

Keywords: side channel analysis, cache analysis, DSA implementation, hyper-threading, sliding window, lattice.

1 Introduction

In 1996, Boneh and Venkatesan studied the following problem: Fix p and n. Having given $(k_i, \mathrm{MSB}_n(xg^{k_i} \bmod p))$ for $1 \leq i \leq d$, where g is a generator of $\mathbb{Z}_p^*$, $\mathrm{MSB}_n(y)$ satisfies $|y - \mathrm{MSB}_n(y)| < \frac{p}{2^{n+1}}$ and d is a polynomial function of $\log p$, find $x \bmod p$. The particular unknown value of x was called a hidden number and the whole problem was named a Hidden Number Problem (HNP). HNP plays an important role in proving security of most significant bits in Diffie-Hellman key agreement and related schemes [7]. In 1999, Nguyen showed a connection between solving HNP and breaking flawed implementations of DSA [13] attacked sooner by Howgrave-Graham and Smart [10]. It was then extended together with Shparlinski in [14]. Their formulation of HNP followed from [7] with a modification allowing them to disclose the private DSA key provided that they know a sufficiently long block of bits of each nonce for several signatures (cf. DSA description in §5.1). The limitation of the method in [14] was that

⋆ The first author was partly supported by the institutional grant MSM 0021620839.

E. Biham and A.M. Youssef (Eds.): SAC 2006, LNCS 4356, pp. 114–133, 2007.

these known bits had to be consecutively placed on the position of either most or least significant bits or they had to occupy a block of bits somewhere in the middle of each nonce. An idea was given on how to overcome this restriction, based on a multidimensional diophantine approximation. Deeper investigation of it then showed that it leads to an unnecessary information loss (cf. remark on §3 below). Furthermore, the method was unaware of direct partial information an attacker may have on the private key itself. A practical cryptanalytic problem may arise, where these factors are limiting. Interestingly, the limiting focus on most significant bits also persisted in other variants derived from the original HNP [21], [6]. In this article, we show an extension of HNP (EHNP) together with a probabilistic algorithm for solving its instances in polynomial time, which relaxes the strict conditions on the form of partial information in HNP significantly. It then allows us to develop, for instance, a successful realistic side channel attack on certain implementations of DSA presented in §5. We consider the EHNP elaboration and the attack presented of independent merit, since we believe that the method presented here can be used as a general tool for solving many other cryptanalytic problems leading to instances of EHNP. Besides (EC)DSA and its variants, we can mention Diffie-Hellman and related schemes such as El Gamal public key encryption, Shamir message passing scheme, and Okamoto conference key sharing studied originally in [7]. Using EHNP allows an analyst to exploit partial information in an "individual bits"-like manner, which is very important with regard to popular side channel attacks. EHNP also allows us to study cryptographic security of individual bits of secrets in these schemes. One can, for instance, extend the results on most significant bits given in [7] to other bits or blocks of bits, as well. It is also possible to implant the ideas elaborated here into other variants of HNP, such as [21], [6]. The practical attack presented, on the other hand, illustrates the significant vulnerability that general widespread applications like SSH server [19] can acquire when using the Pentium 4 HTT processor [11] as a hosting platform.

The rest of the paper is organized as follows: In §2, we review elementary properties of lattices that we use to solve EHNP. We note that instead of a set of diophantine inequalities, we view HNP and its extensions as a special kind of a set of linear congruences of truncated variables. In this way, we get a particular case of the general one studied in [8] (cf. also [4]) which, however, leads to a far easier solution. This idea itself is not new (cf. [16]). However, we show how to exploit it to rigorously prove solvability of (E)HNP without relying on the transforming approximation suggested in [14]. Especially, we exploit the fact that there is a variable (the hidden number) that appears (with a nonzero coefficient) in each congruence from the set.

In §3, we illustrate an evolution of EHNP starting from the approach of [14]. We recall the results of [14] being important for us here in a form of theorems. To stay focused on algorithmic part of the connection in between EHNP and lattices, we omit otherwise very interesting elaboration of distribution conditions

relaxation from the theorems. We slightly generalize the method used there to exploit the partial information in the middle of each nonce. We define HNP-2H problem and show how it reduces to HNP to demonstrate the idea of [14] on how to use the information spread across individual bits. Informally speaking, the authors suggested using certain kind of diophantine approximation to convert these problems to HNP. With such an approach, however, the amount of information needed per nonce grows with a multiple of the number of unknown bit blocks (as already mentioned in [14] and [16]). Contrary to this approach, the EHNP solving method presented here is compact, which means that it does not rely on the conversion causing the unnecessary information loss. In this sense, our method is similar to the one presented by Howgrave-Graham and Smart in [10] which, however, contains a lot of heuristic assumptions in comparison with the results presented here. We hope the whole elaboration in §3 is a useful guideline for a cryptanalyst deciding which method to choose by the kind and amount of information she has.

In §4, we define EHNP, present a lattice-based algorithm for searching candidate solution in polynomial time, and investigate correctness of that solution. To give an illustrative presentation of our ideas, we use certain bounds for lattice problems employed that are not very tight. Actually, we rely only on the well known algorithms of Lenstra, Lenstra, Lovász [12], and Babai [3]. Even with these conservative results, we are able to derive practically acceptable bounds for the EHNP solving algorithm. As being widely understood in the area of lattice problems [9], in practice, however, we may reasonably assume that the algorithms (or their equivalents, cf. §2) will behave much better than what would be guaranteed even by more strict bounds for more matured techniques. Therefore, we suggest an experimental verification of the EHNP instance solvability for a particular cryptanalytic problem, even when it seems to be beyond the estimates guaranteed formally.

In §5, we present a practical side channel attack on certain implementations of DSA (including the OpenSSL one) when running on the Pentium 4 HTT platform [11]. Besides being of an independent merit, it serves as an example of using EHNP approach in a real cryptanalytic case. Finally, we conclude in §6.

2 Preliminaries

The main tool of our approach is a lattice. We define a (full rank) lattice $\mathcal{L}$ in $\mathbb{Q}^d$ as the set of lattice vectors

$$\left\{ \sum_{i=1}^{d} \alpha_i \mathbf{b}_i \mid \alpha_i \in \mathbb{Z} \right\}, \tag{1}$$

where $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{Q}^d$ are linearly independent and are called basis vectors of lattice $\mathcal{L}$. The matrix whose rows are the basis vectors is called basis matrix of lattice $\mathcal{L}$. There is an algorithmic problem connected with the lattices called Approximate Closest Vector Problem (ACVP). Given d linearly independent

basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{Q}^d$ together with $\mathbf{v} \in \mathbb{Q}^d$, the task is to find a lattice vector $\mathbf{W} \in \mathcal{L}$ satisfying $\|\mathbf{v} - \mathbf{W}\| \leq f(d) \min_{\mathbf{A} \in \mathcal{L}} \|\mathbf{v} - \mathbf{A}\|$. The coefficient $f(d)$ is called the approximation factor. The problem is known to be NP-hard for $f(d) = 1$ for any norm [9]. In practical cryptanalytic cases, however, a weaker approximation still suffices. In particular, we will expect there exists a polynomial time algorithm solving ACVP with $f(d) = 2^{\frac{d}{4}}$, which was the basic assumption in [7], [14] (cf. Babai algorithm in [3]). We can get such an algorithm by using various number-theoretic libraries such as NTL [20]. Furthermore, an attacker will probably choose some of the most recent methods like [15]. Note that Algorithm 1 adapts "automatically" for such modification and the estimates of Theorem 5 can be adjusted very easily by a change of the parameter κ_D (cf. the elaboration in §4 for its precise definition).

To simplify the notation later, it is useful to define symbol $|a|_N = \min_{k \in \mathbb{Z}} |a - kN|$ where $N \in \mathbb{N}$ and $a \in \mathbb{Z}$. It is easy to see that

1. $|a + b|_N = 0 \Leftrightarrow a \equiv -b \,(\mathbf{mod}\, N)$
2. $|a + b|_N \leq |a|_N + |b|_N$
3. $|ab|_N \leq |a|_N |b|_N$
4. $|a|_N = \min\{a \,\mathbf{mod}\, N,\ N - (a \,\mathbf{mod}\, N)\}$
5. $|a|_N \leq |a|$

for all $a, b \in \mathbb{Z}$.

As N is a prime throughout the whole paper, $\mathbb{Z}_N$ is regarded as the finite field $\mathbb{Z}_N(+, \cdot, 0, 1)$. Unless stated otherwise, the elements of $\mathbb{Z}_N$ are represented as integers from the set $\{0, \ldots, N-1\}$.

3 On the Way from HNP to EHNP

Definition 1 (Hidden Number Problem). *Let N be a prime and let x, $x \in \mathbb{Z}_N$ be a particular unknown integer that satisfies d congruences*

$$\alpha_i x + \rho_i k_i \equiv \beta_i \,(\mathbf{mod}\, N), \quad 1 \leq i \leq d,$$

where α_i, $\alpha_i \not\equiv 0 \,(\mathbf{mod}\, N)$, ρ_i and β_i, $1 \leq i \leq d$ are known values. The unknown integers k_i satisfy $0 \leq k_i < 2^{\mu}$, $1 \leq i \leq d$, where μ is a known rational constant. The Hidden Number Problem (HNP) is to find x.

Theorem 1 (Solving HNP). *There exists an algorithm running in polynomial time that solves HNP instance, where α_i and ρ_i, $1 \leq i \leq d$ are uniformly and independently distributed on $\langle 1, N-1 \rangle$, with the probability of success*

$$P > 1 - \frac{2^{d\mu}}{(N-1)^{d-1}} \left(1 + 2^{\frac{d+1}{4}} (1+d)^{\frac{1}{2}}\right)^d.$$

Proof. Based on rephrasing the results from [14].

Definition 2 (HNP with Two Holes). *Let N be a prime and let x, $x \in \mathbb{Z}_N$ be a particular unknown integer that satisfies d congruences*

$$\alpha_i x + \rho_{i,1} k_{i,1} + \rho_{i,2} k_{i,2} \equiv \beta_i \ (\mathbf{mod}\, N), \quad 1 \leq i \leq d, \tag{2}$$

where α_i, $\alpha_i \not\equiv 0 \ (\mathbf{mod}\, N)$, $\rho_{i,1}$, $\rho_{i,2}$ and β_i, $1 \leq i \leq d$ are known values. The unknown integers $k_{i,1}$ and $k_{i,2}$ satisfy $0 \leq k_{i,1} < 2^{\mu_1}$ and $0 \leq k_{i,2} < 2^{\mu_2}$, $1 \leq i \leq d$, where μ_1 and μ_2 are known rational constants. The Hidden Number Problem with Two Holes (HNP-2H) is to find x.

Theorem 2 (Dirichlet, [9]). *Let $\alpha \in \mathbb{R}$ and $0 < \varepsilon \leq 1$ be given values. Then there exist $p, q \in \mathbb{Z}$ such that*

$$1 \leq q \leq \frac{1}{\varepsilon} \quad \textit{and} \quad \left| \alpha - \frac{p}{q} \right| < \frac{\varepsilon}{q}.$$

Corollary 1. *Let us be given $A, N \in \mathbb{Z}$ and $B \in \mathbb{R}$ satisfying $B \geq 1$ and $N > 0$. Then there exists λ, $\lambda \in \mathbb{Z}$ such that*

$$1 \leq \lambda \leq B \quad \textit{and} \quad |\lambda A|_N < \frac{N}{B}.$$

Proof. Theorem 2 states there exist $p, q \in \mathbb{Z}$ such that $\left| \frac{A}{N} - \frac{p}{q} \right| < \frac{1}{Bq}$ and $1 \leq q \leq B$. So $|qA|_N \leq |qA - Np| < \frac{N}{B}$. Setting $\lambda = q$ finishes the proof. □

Remark 1. Note that λ promised by Corollary 1 can be easily found in polynomial time using a technique based on the continued fractions expansion (cf. [9], [14]).

Theorem 3 (Solving HNP-2H using Dirichlet's approximation). *There exists an algorithm running in polynomial time that solves HNP-2H, where α_i, $\rho_{i,1}$, and $\rho_{i,2}$, $1 \leq i \leq d$ are uniformly and independently distributed on $\langle 1, N-1 \rangle$, with the probability of success*

$$P > 1 - \frac{(N 2^{\mu_1 + \mu_2})^{\frac{d}{2}}}{(N-1)^{d-1}} \left(4 + 2^{\frac{d+9}{4}} (1+d)^{\frac{1}{2}} \right)^d.$$

Proof. Let $A_i = (\rho_{i,1})^{-1} \rho_{i,2} \,\mathbf{mod}\, N$, $\gamma_i = k_{i,1} + A_i k_{i,2}$, $\alpha_i' = (\rho_{i,1})^{-1} \alpha_i \,\mathbf{mod}\, N$ and $\beta_i' = (\rho_{i,1})^{-1} \beta_i \,\mathbf{mod}\, N$, $1 \leq i \leq d$. The congruences (2) in Definition 2 become

$$\alpha_i' x + \gamma_i \equiv \beta_i' \ (\mathbf{mod}\, N), \quad 1 \leq i \leq d. \tag{3}$$

Given any $B \in \mathbb{R}, B \geq 1$, we can find non-zero integers $\lambda_{i,B}$ satisfying $|\lambda_{i,B} A_i| < \frac{N}{B}$ and $1 \leq \lambda_{i,B} \leq B$ for $1 \leq i \leq d$. It holds

$$|\lambda_{i,B} \gamma_i|_N = |\lambda_{i,B} k_{i,1} + \lambda_{i,B} A_i k_{i,2}|_N \leq |\lambda_{i,B}|_N k_{i,1} + |\lambda_{i,B} A_i|_N k_{i,2} < B 2^{\mu_1} + \frac{N}{B} 2^{\mu_2}.$$

The choice $B_{min} = N^{\frac{1}{2}} 2^{\frac{\mu_2 - \mu_1}{2}}$ minimizes the upper bound $B 2^{\mu_1} + \frac{N}{B} 2^{\mu_2}$.

We convert HNP-2H to HNP by setting $k'_i = \left(\lambda_{i,B_{min}}\gamma_i + \lfloor N^{\frac{1}{2}}2^{\frac{\mu_1+\mu_2+2}{2}}\rfloor\right) \mathbf{mod}\, N$, $k'_i < N^{\frac{1}{2}}2^{\frac{\mu_1+\mu_2+4}{2}}$. After several modifications of (3), we obtain congruences in one unknown variable k'_i per congruence, i.e.

$$\begin{aligned}(\lambda_{i,B_{min}}\alpha'_i)\,x + \lambda_{i,B_{min}}\gamma_i &\equiv \lambda_{i,B_{min}}\beta'_i \quad (\mathbf{mod}\, N),\\ (\lambda_{i,B_{min}}\alpha'_i)\,x + k'_i &\equiv \lambda_{i,B_{min}}\beta'_i + \left\lfloor N^{\frac{1}{2}}2^{\frac{\mu_1+\mu_2+2}{2}}\right\rfloor \quad (\mathbf{mod}\, N),\\ \alpha''_i x + k'_i &\equiv \beta''_i \quad (\mathbf{mod}\, N), \quad 1 \le i \le d\end{aligned}$$

defining an instance of HNP. Let $\mu' \in \mathbb{Q}$ be such that $k'_i < 2^{\mu'} \le N^{\frac{1}{2}}2^{\frac{\mu_1+\mu_2+4}{2}}$. By Theorem 1, such a problem can be solved in polynomial time with the probability

$$\begin{aligned}P &> 1 - \frac{2^{d\mu'}}{(N-1)^{d-1}}\left(1 + 2^{\frac{d+1}{4}}(1+d)^{\frac{1}{2}}\right)^d \ge 1 - \frac{N^{\frac{d}{2}}2^{\frac{(\mu_1+\mu_2+4)d}{2}}}{(N-1)^{d-1}}\left(1 + 2^{\frac{d+1}{4}}(1+d)^{\frac{1}{2}}\right)^d =\\ &= 1 - \frac{(N2^{\mu_1+\mu_2})^{\frac{d}{2}}}{(N-1)^{d-1}}2^{2d}\left(1 + 2^{\frac{d+1}{4}}(1+d)^{\frac{1}{2}}\right)^d = 1 - \frac{(N2^{\mu_1+\mu_2})^{\frac{d}{2}}}{(N-1)^{d-1}}\left(4 + 2^{\frac{d+9}{4}}(1+d)^{\frac{1}{2}}\right)^d.\end{aligned}$$

□

It is correct to interpret Theorem 3 as saying that we need roughly twice as many information bits to solve HNP-2H compared to the plain HNP case [16]. This is caused by the transforming approximation and it is generally independent on the technique used to solve the transformed HNP. If we continue further this way to define HNP with more "holes" (HNP-xH), we will need to use a multidimensional transforming approximation based e.g. on the scope of the multidimensional Dirichlet's theorem and lattice reduction techniques [9]. What we obtain is then a conjecture stated in [16] that we need at least x-times as many information bits to solve HNP-xH compared to the plain HNP case. However, using the algorithmic and mainly the proving strategies described bellow, it turns out that such a conjecture does not hold generally. We can see it, for instance, by normalizing the probability estimations given above and bellow under the assumption that we have an access to an oracle solving the Closest Vector Problem with an arbitrary precision for the maximum norm. We omit this demonstration here, since it is beyond the scope of the paper. Therefore, the conjecture of [16] is not a property of HNP itself, it is a property of a particular method for solving HNP instead. On the other hand, this is not the only one selection criterion. As is demonstrated bellow, our method does not suffer from the expensive transforming approximation, but, on the other hand, it is more sensitive to the approximation factor of the particular algorithm used for solving the Approximate Closest Vector Problem.

Theorem 4 (Solving HNP-2H as a special case of EHNP). *There exists an algorithm running in polynomial time that solves HNP-2H, where α_i, $\rho_{i,1}$, and $\rho_{i,2}$, $1 \le i \le d$ are uniformly and independently distributed on $\langle 1, N-1\rangle$, with the probability of success*

$$P > 1 - \frac{2^{(\mu_1+\mu_2)d}}{N^{d-1}}\left(1 + 2^{\frac{3d+1}{4}}(1+2d)^{\frac{1}{2}}\right)^{2d+1}.$$

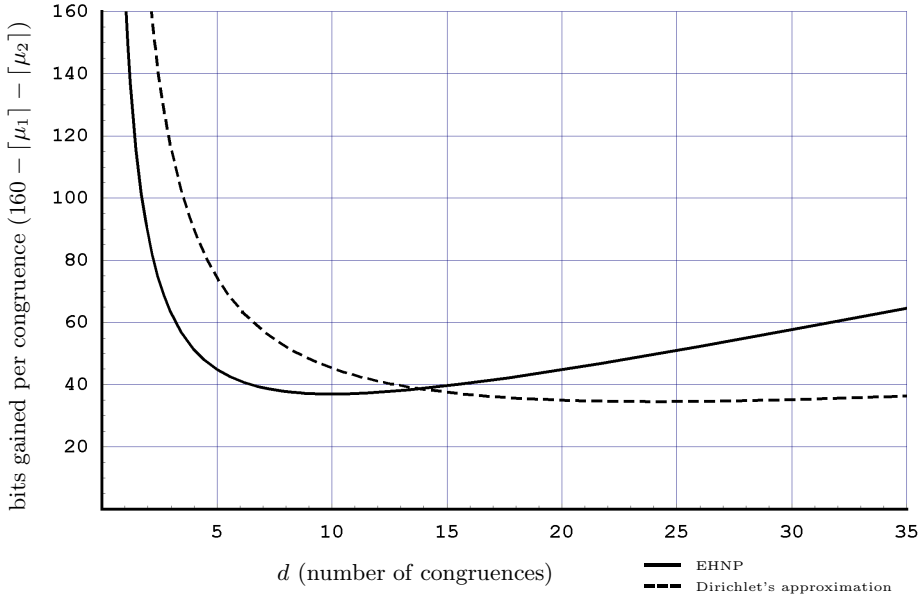

Fig. 1. Dirichlet's approximation vs. EHNP algorithm solving HNP-2H

Proof. The algorithm arises from the solution of EHNP defined and solved in the following section, which HNP-2H is a special case of.

Figure 1 shows a comparison of HNP-2H solving using Dirichlet's approximation and EHNP for 160 bits long modulus N. The graph lines connect the boundary points corresponding to probably solvable combinations of the number of bits gained per congruence and the number of congruences (e.g. signatures of DSA). The EHNP approach is provably preferable in cases with high amount of information available for only few congruences. These are the cases arising, for instance, in fault side channel attacks, where the device may get burnt soon, however, revealing a huge amount of information before being irreversibly damaged. In practice, we may expect a broader preference of EHNP, since the approximation factor will be probably much better than the estimate we used.

4 Extended Hidden Number Problem

Definition 3 (Extended Hidden Number Problem). *Let N be a prime and let x, $x \in \mathbb{Z}_N$, be a particular unknown integer such that*

$$x = \bar{x} + \sum_{j=1}^{m} 2^{\pi_j} x_j, \tag{4}$$

where the integers $\bar{x}$ and π_j, $1 \le j \le m$ are known. The unknown integers x_j satisfy $0 \le x_j < 2^{\nu_j}$, where ν_j are known rational constants $1 \le j \le m$. Furthermore, let us be given d congruences

$$\alpha_i \sum_{j=1}^{m} 2^{\pi_j} x_j + \sum_{j=1}^{l_i} \rho_{i,j} k_{i,j} \equiv \beta_i - \alpha_i \bar{x} \,(\mathbf{mod}\, N), \quad 1 \le i \le d, \tag{5}$$

where α_i, $\alpha_i \not\equiv 0 \,(\mathbf{mod}\, N)$, $1 \le i \le d$, π_j, $1 \le j \le m$, $\rho_{i,j}$, $1 \le i \le d$, $1 \le j \le l_i$ and β_i, $1 \le i \le d$ are known values. The unknown integers $k_{i,j}$ satisfy $0 \le k_{i,j} < 2^{\mu_{i,j}}$, where $\mu_{i,j}$ are known, $1 \le i \le d, 1 \le j \le l_i$. We define $\tau = \sum_{j=1}^{m} \nu_j$, $\xi_i = \sum_{j=1}^{l_i} \mu_{i,j}$, $1 \le i \le d$ and $\xi = \sum_{i=1}^{d} \xi_i$.

The Extended Hidden Number Problem (EHNP) is to find (the hidden number) x and its instance is represented by

$$\left(\bar{x}, N, \{\pi_j, \nu_j\}_{j=1}^{m}, \left\{\alpha_i, \{\rho_{i,j}, \mu_{i,j}\}_{j=1}^{l_i}, \beta_i\right\}_{i=1}^{d}\right). \tag{6}$$

Definition 4 (Lattice $\mathcal{L}(\mathcal{I}, \delta)$ and its basis matrix). *For $\delta > 0$ and a given instance $\mathcal{I} = \mathcal{I}_{EHNP}$ of the EHNP we define $\mathcal{L}(\mathcal{I}, \delta)$ as the lattice spanned by the rows of the matrix*

$$\mathbf{B} = \mathbf{B}(\mathcal{I}, \delta) = \begin{pmatrix} N \cdot \mathbf{I}_d & \emptyset & \emptyset \\ \mathbf{A} & \mathbf{X} & \emptyset \\ \boldsymbol{\rho}_1^T & & \\ \ddots & \emptyset & \mathbf{K} \\ \boldsymbol{\rho}_d^T & & \end{pmatrix} \in \mathbb{Q}^{D \times D},$$

where we define integers $L = \sum_{i=1}^{d} l_i$ and $D = d + m + L$, vectors

$$\boldsymbol{\rho}_i = (\rho_{i,1}, \ldots, \rho_{i,l_i}) \in \mathbb{Z}^{l_i}, \quad 1 \le i \le d$$

and matrices

$$\mathbf{A} = (a_{j,i})_{1 \le i \le d, 1 \le j \le m} \in \mathbb{Z}^{m \times d}, \text{ where } a_{j,i} = \alpha_i 2^{\pi_j}$$

$$\mathbf{X} = \mathbf{diag}\left(\frac{\delta}{2^{\nu_1}}, \ldots, \frac{\delta}{2^{\nu_m}}\right) \in \mathbb{Q}^{m \times m}$$

$$\mathbf{K} = \mathbf{diag}\left(\frac{\delta}{2^{\mu_{1,1}}}, \ldots, \frac{\delta}{2^{\mu_{1,l_1}}}, \frac{\delta}{2^{\mu_{2,1}}}, \ldots, \frac{\delta}{2^{\mu_{2,l_2}}}, \ldots, \frac{\delta}{2^{\mu_{d,1}}} \ldots, \frac{\delta}{2^{\mu_{d,l_d}}}\right) \in \mathbb{Q}^{L \times L}.$$

Lemma 1 (Short vectors in $\mathcal{L}$). *Let $\mathcal{I}$ be an instance of the EHNP, where α_i, $1 \le i \le d$ and $\rho_{i,j}$, $1 \le i \le d$, $1 \le j \le l_i$ are uniformly and independently*

Algorithm 1. Finding a solution candidate for EHNP

Input: Instance $\mathcal{I}$ of EHNP
Output: Solution candidate $z \in \mathbb{Z}_N$

1: $\kappa_D \leftarrow \frac{2^{\frac{D}{4}}(m+L)^{\frac{1}{2}}+1}{2}$
2: Choose δ such that $0 < \kappa_D\delta < 1$
3: $\mathbf{v} \leftarrow \left((\beta_1 - \alpha_1\bar{x}) \bmod N, \ldots, (\beta_d - \alpha_d\bar{x}) \bmod N, \frac{\delta}{2}, \ldots, \frac{\delta}{2}\right)$
4: find $\mathbf{W} \in \mathcal{L} = \mathcal{L}(\mathcal{I}, \delta)$, $\mathbf{W} = (W_1, \ldots, W_D)$ such that $\|\mathbf{W} - \mathbf{v}\| \leq 2^{\frac{D}{4}} \min_{\mathbf{B}\in\mathcal{L}} \|\mathbf{v} - \mathbf{B}\|$ *//in polynomial time (§2)*
5: **for** $j = 1$ **to** m **do**
6: $\quad x'_j \leftarrow \frac{W_{d+j}2^{\nu_j}}{\delta}$ *//$x'_j \in \mathbb{Z}$*
7: **end for**
8: $z \leftarrow \bar{x} + \sum_{j=1}^{m} 2^{\pi_j} x'_j \bmod N$
9: **return** z

distributed on $\langle 1, N-1\rangle$. *Let* $\delta, \kappa_D \in \mathbb{Q}$ *be such that* $0 < \delta$, $0 < \kappa_D$ *and* $\kappa_D\delta < 1$. *Then with the probability*

$$P > 1 - \frac{(2\kappa_D)^{L+m}\, 2^{\tau+\xi}}{N^d} \tag{7}$$

for each vector $\boldsymbol{\Delta} \in \mathcal{L} = \mathcal{L}(\mathcal{I}, \delta)$ *with coordinates* $\mathbf{c} = (e_1, \ldots, e_d, y_1, \ldots, y_m, t_{1,1}, \ldots, t_{1,l_1}, \ldots, \ldots, t_{d,1}, \ldots, t_{d,l_d})$ *w.r.t. basis* $\mathbf{B} = \mathbf{B}(\mathcal{I}, \delta)$ *(i. e.* $\boldsymbol{\Delta} = \mathbf{cB}$*), satisfying* $\|\boldsymbol{\Delta}\|_\infty < \kappa_D\delta$

(i) there exists (a witness index) w, $1 \leq w \leq d$ *such that*

$$t_{w,j} \equiv 0 \ (\bmod N), \quad 1 \leq j \leq l_w, \tag{8}$$

(ii) $\sum_{j=1}^{m} 2^{\pi_j} y_j \equiv 0 \ (\bmod N)$ *holds,*
(iii) $\sum_{j=1}^{l_i} \rho_{i,j} t_{i,j} \equiv 0 \ (\bmod N), 1 \leq i \leq d$ *holds.*

Proof. To be found in Appendix.

Theorem 5 (Correctness of the algorithm solving EHNP). *Let* x *be the solution of EHNP specified by the instance* $\mathcal{I} = \mathcal{I}_{EHNP}$ *where* N *is prime,* α_i, $1 \leq i \leq d$ *and* $\rho_{i,j}$, $1 \leq i \leq d$, $1 \leq j \leq l_i$ *are uniformly and independently distributed on* $\langle 1, N-1\rangle$. *Then with the probability*

$$P > 1 - \frac{(2\kappa_D)^{L+m}\, 2^{\tau+\xi}}{N^d}, \tag{9}$$

where $\kappa_D = \frac{2^{\frac{D}{4}}(m+L)^{\frac{1}{2}}+1}{2}$, *Algorithm 1 returns the correct particular solution of instance* $\mathcal{I}$.

Proof. Let $\delta \in \mathbb{Q}$ be such that $0 < \kappa_D \delta < 1$. By Definition 3 there exists vector $\mathbf{h} = (c_1, \ldots, c_d,\ x_1, \ldots, x_m,\ k_{1,1}, \ldots, k_{1,l_1},\ \ldots,\ \ldots,\ k_{d,1}, \ldots, k_{d,l_d}) \in \mathbb{Z}^D$ such that

$$c_i N + \alpha_i \sum_{j=1}^{m} 2^{\pi_j} x_j + \sum_{j=1}^{l_i} \rho_{i,j} k_{i,j} = \beta_i - \alpha_i \bar{x}, \quad 1 \le i \le d, \tag{10}$$

$$0 \le x_j < 2^{\nu_j}, \quad 1 \le j \le m, \tag{11}$$

$$0 \le k_{i,j} < 2^{\mu_{i,j}}, \quad 1 \le i \le d,\ 1 \le j \le l_i \tag{12}$$

hold. Let $\mathbf{B} = \mathbf{B}(\mathcal{I}, \delta)$ be the matrix defined in Definition 4 and let

$$\mathbf{H} = \mathbf{hB} \in \mathcal{L},\ \mathcal{L} = \mathcal{L}(\mathcal{I}, \delta),$$
$$\mathbf{v} = ((\beta_1 - \alpha_1 \bar{x}) \bmod N, \ldots, (\beta_d - \alpha_d \bar{x}) \bmod N, \frac{\delta}{2}, \ldots, \frac{\delta}{2}) \in \mathbb{Z}^D.$$

Since the vector $\mathbf{H} - \mathbf{v}$ is equal to

$$\Big(0, \ldots, 0, \delta\tfrac{x_1}{2^{\nu_1}} - \tfrac{\delta}{2}, \ldots, \delta\tfrac{x_m}{2^{\nu_m}} - \tfrac{\delta}{2}, \delta\tfrac{k_{1,1}}{2^{\mu_{1,1}}} - \tfrac{\delta}{2}, \ldots, \delta\tfrac{k_{1,l_1}}{2^{\mu_{1,l_1}}} - \tfrac{\delta}{2}, \ldots,$$
$$\ldots, \delta\tfrac{k_{d,1}}{2^{\mu_{d,1}}} - \tfrac{\delta}{2}, \ldots, \delta\tfrac{k_{d,l_d}}{2^{\mu_{d,l_d}}} - \tfrac{\delta}{2}\Big),$$

and the bounds (11), (12) hold, we can write

$$\|\mathbf{v} - \mathbf{H}\|_\infty < \frac{\delta}{2}.$$

A lattice vector $\mathbf{W}$ found in step 4 of the algorithm satisfies

$$\|\mathbf{v} - \mathbf{W}\| \le 2^{\frac{D}{4}} \min_{\mathbf{A} \in \mathcal{L}} \|\mathbf{v} - \mathbf{A}\| \le 2^{\frac{D}{4}} \|\mathbf{v} - \mathbf{H}\| < \frac{2^{\frac{D}{4}} \delta (m + L)^{\frac{1}{2}}}{2}. \tag{13}$$

Let $\boldsymbol{\Delta} = \mathbf{H} - \mathbf{W} \in \mathcal{L}$. Since $\|\mathbf{a}\|_\infty \le \|\mathbf{a}\|$ for all $\mathbf{a} \in \mathbb{Z}^D$, by triangle inequality we have

$$\|\boldsymbol{\Delta}\|_\infty \le \|\mathbf{H} - \mathbf{v}\|_\infty + \|\mathbf{v} - \mathbf{W}\| < \frac{\delta}{2} + \frac{2^{\frac{D}{4}} \delta (m + L)^{\frac{1}{2}}}{2} = \kappa_D \delta. \tag{14}$$

Let $\mathbf{w}$, $\boldsymbol{\gamma}$ be the coordinate vectors of $\mathbf{W}$, $\boldsymbol{\Delta}$, respectively, w.r.t. basis $\mathbf{B}$ and

$$\mathbf{w} = (c'_1, \ldots, c'_d, x'_1, \ldots, x'_m, k'_{1,1}, \ldots, k'_{1,l_1}, \ldots, \ldots, \ldots, k'_{d,1}, \ldots, k'_{d,l_d})$$
$$\boldsymbol{\gamma} = (e_1, \ldots, e_d, y_1, \ldots, y_m, t_{1,1}, \ldots, t_{1,l_1}, \ldots, \ldots, \ldots, t_{d,1}, \ldots, t_{d,l_d})$$

and $z = \bar{x} + \sum_{j=1}^{m} 2^{\pi_j} x'_j$ be the candidate returned by Algorithm 1. Since $\mathbf{B}$ is nonsingular, $\boldsymbol{\gamma} = \mathbf{h} - \mathbf{w}$. Then with the probability greater than $1 - \frac{(2\kappa_D)^{L+m} 2^{\tau+\xi}}{N^d}$, guaranteed by Lemma 1, it holds

$$x - z \equiv \left(\bar{x} + \sum_{j=1}^{m} 2^{\pi_j} x_j\right) - \left(\bar{x} + \sum_{j=1}^{m} 2^{\pi_j} x'_j\right) \equiv$$
$$\equiv \sum_{j=1}^{m} 2^{\pi_j} (x_j - x'_j) \equiv \sum_{j=1}^{m} 2^{\pi_j} y_j \equiv 0 \ (\bmod N).$$

Finally, since $x, z \in \mathbb{Z}_N$, we have $x = z$. □

The distribution conditions in Theorem 5 can be, in a particular cryptanalytic case, further relaxed using techniques like in [14]. For verification of the attack in §5, however, we decided to use an experimental approach instead.

5 Real Scenario - Digital Signature Algorithm

5.1 DSA Key Disclosure Problem

Let us briefly recall the Digital Signature Algorithm (DSA) [1]. The public parameters are specified by triplet (p, q, g), where p and q are prime numbers satisfying $q|p-1$ and $g \in \mathbb{Z}_p^*$ is a generator of the subgroup of order q in $\mathbb{Z}_p^*$. The second revision of [1] requires $2^{1023} < p < 2^{1024}$ and $2^{159} < q < 2^{160}$. The private key $x \in \mathbb{Z}_q$ is chosen uniformly at random from $\mathbb{Z}_q$ and the corresponding public key $y \in \mathbb{Z}_p$ is computed as $y = g^x \bmod p$. The couple (x, y) is called the DSA key pair.

To create a signature (r, s) of a message $m \in \{0,1\}^*$, the owner of the private key x first generates a random number $k \in \mathbb{Z}_q^*$, which is usually referred to as a nonce (number used once). Then she computes $r = \left(g^k \bmod p\right) \bmod q$ and $s = k^{-1}(h(m) + xr) \bmod q$, where h is the hash function SHA-1 (see [2]) and $k^{-1}k \equiv 1 \ (\bmod q)$.

To verify the signature pair (r, s) of a message m, having checked that $0 < r < q$ and $0 < s < q$, one computes $w = s^{-1} \bmod q$, $u_1 = h(m)w \bmod q$, $u_2 = rw \bmod q$ and $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$. She accepts the message signature if and only if $v = r$.

Definition 5 (DSA-KDP problem). *Let (p, q, g) be public DSA parameters and (x, y) DSA key pair. Let*

$$r_i = \left(g^{k_i} \bmod p\right) \bmod q, \quad 1 \le i \le d \tag{15}$$

$$s_i = k_i^{-1} \left(h(m_i) + xr_i\right) \bmod q, \quad 1 \le i \le d \tag{16}$$

be known signature pairs of known hashed messages $h(m_i)$. Suppose an additional information about the private key x and the nonces k_i is known, as well, i.e.

$$x = \bar{x} + \sum_{j=1}^{m} 2^{\pi_j} x_j, \ 0 \le x_j < 2^{\nu_j}, \quad 1 \le j \le m \tag{17}$$

$$k_i = \bar{k}_i + \sum_{j=1}^{l_i} 2^{\lambda_{i,j}} k_{i,j}, \ 0 \le k_{i,j} < 2^{\mu_{i,j}}, \quad 1 \le i \le d, 1 \le j \le l_i, \tag{18}$$

where $\bar{x}$, $\{\pi_j, \nu_j\}_{j=1}^{m}$, $\left\{\bar{k}_i, \{\lambda_{i,j}, \mu_{i,j}\}_{j=1}^{l_i}\right\}_{i=1}^{d}$ are known integers satisfying

$$\bar{x} \in \mathbb{Z}_q, \ \bar{k}_i \in \mathbb{Z}_q, \quad 1 \le i \le d$$
$$2^{\pi_j} \in \mathbb{Z}_q, \quad 1 \le j \le m, \ 2^{\lambda_{i,j}} \in \mathbb{Z}_q, \quad 1 \le i \le d, \ 1 \le j \le l_i$$
$$2^{\nu_j} \in \mathbb{Z}_q, \quad 1 \le j \le m, \ 2^{\mu_{i,j}} \in \mathbb{Z}_q, \quad 1 \le i \le d, \ 1 \le j \le l_i$$

DSA key disclosure problem (DSA-KDP) is to find the private key x and its instance $\mathcal{I}_{DSA}$ is represented by

$$\left(\bar{x}, q, \{r_i, s_i, h(m_i)\}_{i=1}^{d}, \{\pi_j, \nu_j\}_{j=1}^{m}, \left\{\bar{k}_i, \{\lambda_{i,j}, \mu_{i,j}\}_{j=1}^{l_i}\right\}_{i=1}^{d}\right).$$

Lemma 2 (Transition from DSA-KDP to EHNP). *Let x be the particular solution of the DSA-KDP problem specified by the instance $\mathcal{I}_{DSA}$. Let*

$$\begin{aligned} N &= q, \\ \alpha_i &= r_i, \quad 1 \le i \le d \\ \rho_{i,j} &= \left(-s_i 2^{\lambda_{i,j}}\right) \bmod N, \quad 1 \le i \le d, 1 \le j \le l_i, \\ \beta_i &= \left(s_i \bar{k}_i - h(m_i)\right) \bmod N, \quad 1 \le i \le d. \end{aligned}$$

Then x can be found as the solution of EHNP specified by the instance

$$\left(\bar{x}, N, \{\pi_j, \nu_j\}_{j=1}^{m}, \left\{\alpha_i, \{\rho_{i,j}, \mu_{i,j}\}_{j=1}^{l_i}, \beta_i\right\}_{i=1}^{d}\right). \tag{19}$$

Proof. The lemma follows directly when (17) and (18) are substituted into (16) in Definition 5. □

5.2 Hyper-threading Technology

In [18], the author explores a side channel hidden in certain processors design employing the Hyper-Threading Technology (HTT). The processors affected are Intel Pentium 4, Mobile Pentium 4 and Xeon.

The size of L1 cache memory in hyper-threaded Pentium 4 is 8 KB (or 16 KB)[1]. It is divided into 32 cache address classes consisting of 4 (or 8) cache lines of 64 bytes. When a memory line is requested by a process, the processor first checks whether the line is already loaded in L1 cache in the corresponding cache class. In case it is, we say a cache hit occurs, contrary to a cache miss when the line has to be loaded to L1 cache. Therefore, a cache miss takes a much longer time than a cache hit and that can be recognized by the process.

A hyper-threaded Pentium 4 multiplexes two independent instruction streams over one set of execution resources creating two virtual processing cores that share certain physical resources, namely the L1 cache memory. This is an architectural design that leaves the Confinement Problem [5] unsolved leading to a vital side channel. To the operating system, this platform appears as two logical CPUs, thus it can schedule two processes to be run at the same time. One process cannot read the data of the other process, however, it can determine whether the process running on the second virtual core used certain line of the L1 cache or not by measuring the amount of time it takes to repeatedly read several data blocks from the same cache address class.

In this way, we get a side information which can be used to discriminate between two different operations performed by the process being spied [17], [18].

[1] Depending on actual type.

When the two operations are different enough with respect to the memory access they induce, we can easily identify them basing on a different "footprint" left in the access time measurements (cf. Figure 2).

In [18], this side information was used to break an implementation of RSA scheme. Here, we show that by using the EHNP approach described before, we can break certain implementation of DSA algorithm, as well. In practice, this attack threatens, for instance, applications like SSH [19], when the server runs on an unsecured HTT platform and uses DSA for the server authentication. When the attacker logs on the server, she can run the spy process on one processor core, while she opens another SSH session with the server, hoping that it will run on the second core. In this way, she gains the side information about DSA signature computation when the server authenticates itself for the newly opened connection. Collecting several such measurements, she can get enough information to be able to solve the associated EHNP. From here, she gets the server's private key allowing her to impersonate the server.

5.3 Sliding Window Exponentiation

An OpenSSL-based SSH server uses the sliding window (SW) exponentiation algorithm (cf. Algorithm 2)[2] in the process of DSA authentication when computing $r = g^k \bmod p$. Two operations to be discriminated on the HTT platform by the aforesaid technique are squaring (S) and multiplication (M). Being able to identify the SW algorithm execution, the attacker obtains a sequence $\mathcal{S} \in \{S, M\}^*$.

To convert $\mathcal{S}$ to an information suitable for EHNP, one sets $k' = 0$ as the first approximation of the nonce k. Then, she takes the next operation from $\mathcal{S}$ and multiplies k' by 2 if the operation is S or adds 1 and adds a new "hole" for M. Finally, the holes of zero length are filtered out from the output sequence. This procedure, described by Algorithm 3, outputs the decomposition $k = k' + \sum_{j=1}^{l} 2^{\lambda_j} k_j$, $0 \leq k_j < 2^{\mu_j}$. On average case, the algorithm provides us with 78 known bits separated by 31 holes for the exponent size of 160 bits with the sliding window length set to 4 (to match its definition in OpenSSL 0.9.7e).

In Figure 2, we can see the execution of SW algorithm from the spy process viewpoint. The rows, each representing one of 32 memory classes in L1 cache, change color from white standing for a very fast read operation, to black for slow data reloads. To emphasize the different footprints, the extra top row displays the average gray level for the selected cache classes (in our case 24th, 25th and 26th class). This row allows us to identify squaring and multiplication operations easily.

5.4 Practical Experiments

Algorithm 1 was implemented in C++ employing Shoup's NTL library [20]. Several experiments were run for different number of signatures d. Each experiment consisted of 10 instances of DSA-KDP with random public parameters,

[2] Valid for the versions up to 0.9.7g. As from version 0.9.7h, OpenSSL uses fixed window modular exponentiation by default for RSA, DSA, and DH private key operations to prevent cache timing attacks.

Algorithm 2. Sliding window (SW) exponentiation; s is the SW length

Input: g, $e = (e_t e_{t-1} \dots e_0)_2$, $e_t = 1$, $s \geq 1$
Output: g^e

1: $g_1 \leftarrow g$, $g_2 \leftarrow g^2$
2: **for** $i = 1$ to $2^{s-1} - 1$ **do**
3: $\quad g_{2i+1} \leftarrow g_{2i-1} g_2$
4: **end for**
5: $A \leftarrow 1$, $i \leftarrow t$
6: **while** $i \geq 0$ **do**
7: $\quad$ **if** $e_i = 0$ **then**
8: $\quad\quad A \leftarrow A^2$ *//squaring*
9: $\quad\quad i \leftarrow i - 1$
10: $\quad$ **else**
11: $\quad\quad$ find longest string $(e_i e_{i-1} \dots e_l)$ such that $i - l + 1 \leq s$ and $e_l = 1$
12: $\quad\quad A \leftarrow A^{2^{i-l+1}} g_{(e_i e_{i-1} \dots e_l)_2}$ *//$i - l + 1$ squarings, 1 multiplication*
13: $\quad\quad i \leftarrow l - 1$
14: $\quad$ **end if**
15: **end while**
16: **return** A

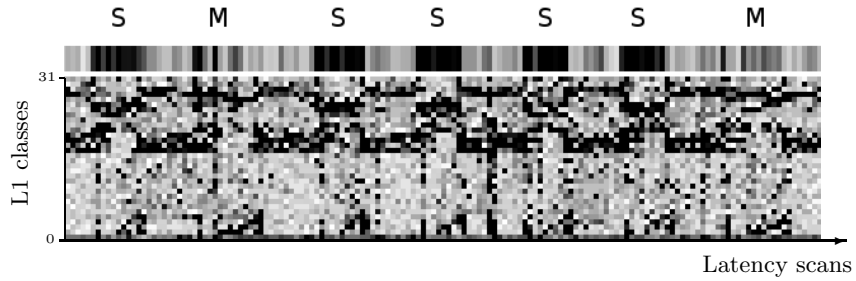

Fig. 2. SW exponentiation observed through the side channel of L1 cache

random key pair and the signature pairs for d random messages. The results of the experiments are displayed in Figures 3 and 4. The computing platform employed was running GNU/Linux Debian on AMD Opteron 844. We used the side channel emulation in these computations. Its real existence and usability was successfully verified by technical experiments with an SSH server powered by OpenSSL 0.9.7e on an unprotected Pentium 4 HTT platform, as well.

The bases were reduced by LLL reduction with `delta = 0.99` (`LLL_XD()`, marked "LLL"). If such reduction did not lead to the key disclosure, stronger Block Korkin-Zolotarev reduction with Givens rotations (`G_BKZ_XD()`, marked "BKZ") was employed with `delta = 0.99`, `BlockSize = 40` and `Prune = 15`.

We ran several experiments with random DSA-KDP instances with the size of DSA prime q set to 256 bits (which is the highest size allowed by the draft

Algorithm 3. Conversion of sequence from $\{S, M\}^*$ to the decomposition of k; s is the SW length; (N is the upper bound on k)

Input: $\mathcal{S} \in \{S, M\}^*$, $s \geq 1$, $(N > 1)$
Output: $\bar{k}, l, \{\lambda_j, \mu_j\}_{j=1}^{l}$

$\bar{k} \leftarrow 0$, $L \leftarrow 0$, $\Lambda_0 \leftarrow 1$, $w_0 \leftarrow s - 1$ *//L, Λ, and w are internal only*
$A \leftarrow \text{first}(\mathcal{S})$
repeat
  if $A = S$ **then**
    $\bar{k} \leftarrow 2\bar{k}$
    for $j = 0$ to L **do**
      $\Lambda_j \leftarrow \Lambda_j + 1$ *//shift existing holes*
    end for
  else
    $\bar{k} \leftarrow \bar{k} + 1$
    $L \leftarrow L + 1$, $\Lambda_L \leftarrow 1$
    $w_L \leftarrow \min(s - 1, \Lambda_{L-1} + w_{L-1} - s - 1)$ *//new, possibly empty, hole*
  end if
until $A \leftarrow \text{next}(\mathcal{S})$
if N is defined **and** $\Lambda_1 + w_1 > \lceil \log_2 N \rceil$ **then**
  $w_1 = \lceil \log_2 N \rceil - \Lambda_1$ *//adjust the first hole*
end if
$l \leftarrow 0$
for $j = 1$ to L **do**
  if $w_j > 0$ **then**
    $l \leftarrow l + 1$, $\lambda_l \leftarrow \Lambda_j$, $\mu_l \leftarrow w_j$ *//keep the nonempty holes*
  end if
end for

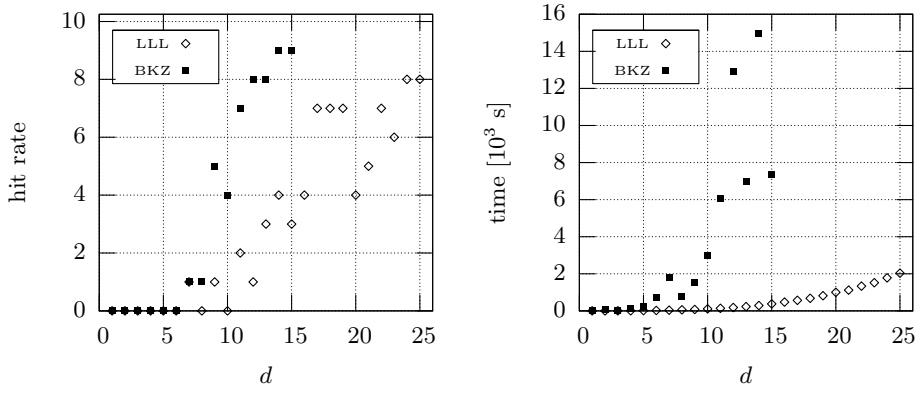

Fig. 3. The hit rate and the average duration of EHNP algorithm solving 10 random instances of DSA-KDP, each derived from a simulated side channel leakages during the signing operation with $\lceil \log_2 q \rceil = 160$

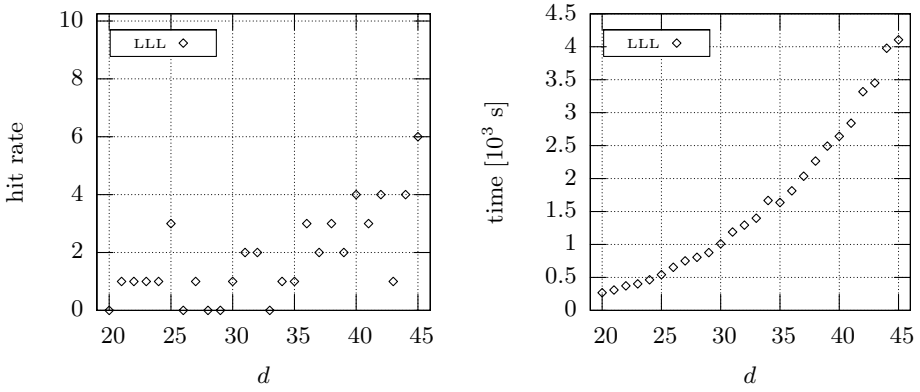

Fig. 4. Hit rate and the average duration of EHNP algorithm solving 10 random instances of DSA-KDP, each derived from a simulated side channel leakages during the signing operation with $\lceil \log_2 q \rceil = 256$. Blocks of known bits shorter than 5 in length are ignored (to reduce running time)

of Third revision of FIPS 186). The dimension of lattices associated is higher resulting in longer running time, however, as shown in Figure 4, the private key can be extracted for these instances, as well.

6 Conclusion

The Hidden Number Problem (HNP) was originally presented as a tool for proving cryptographic security of Diffie-Hellman and related schemes [7]. It was then shown in [13] and [14] that HNP can be used for solving cryptanalytic problems arising around DSA, as well. However, it still retained certain properties that limited its suitability for real cryptanalytic attacks. In this article, we showed how to overcome these limitations by formulation of the Extended Hidden Number Problem (EHNP) that covers significantly broader area of practical cases. Algorithm for solving EHNP together with its formal analysis were presented. We employed EHNP to develop a practically feasible side channel attack on the OpenSSL implementation of DSA. This part of the paper is of independent merit showing an inherent insecurity of the Pentium 4 HTT processor platform for applications like SSH. For instance, having observed only 6 authentications to the OpenSSL-powered SSH server, an attacker was able to disclose the server's private key.

Acknowledgment

The authors would like to thank Colin Percival for kindly providing them the source code for his experiments and the anonymous referees for their valuable

comments. The first author thanks Jozef Juríček for helpful discussions covering several topics in probability theory. The second author appreciates eBanka a.s. supporting these research activities.

References

1. Digital signature standard. National Institute of Standards and Technology, Washington (Note: Federal Information Processing Standard 186-2) (2000), URL: http://csrc.nist.gov/publications/fips/
2. Secure hash standard. National Institute of Standards and Technology, Washington (Note: Federal Information Processing Standard 180-2) (2002), URL: http://csrc.nist.gov/publications/fips/
3. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. In: Mehlhorn, K. (ed.) STACS 85. LNCS, vol. 182, pp. 13–20. Springer, Heidelberg (1984)
4. Bellare, M., Goldwasser, S., Micciancio, D.: "Pseudo-Random" number generation within cryptographic algorithms: The DDS case. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 277–291. Springer, Heidelberg (1997)
5. Bishop, M.: Computer Security: Art and Science. Addison-Wesley, Reading (2003)
6. Boneh, D., Halevi, S., Howgrave-Graham, N.: The modular inversion hidden number problem. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 36–51. Springer, Heidelberg (2001)
7. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 129–142. Springer, Heidelberg (1996)
8. Frieze, A.M., Hastad, J., Kannan, R., Lagarias, J.C., Shamir, A.: Reconstructing truncated integer variables satisfying linear congruences. SIAM Journal of Computing 17(2), 262–280 (1988)
9. Groetschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, 2nd edn. Springer, Heidelberg (1993)
10. Howgrave-Graham, N.-A., Smart, N.P.: Lattice attacks on digital signature schemes. Design, Codes and Cryptography 23, 283–290 (2001)
11. Intel Corporation. Intel(R) Pentium(R) 4 Processor supporting Hyper-Threading Technology. URL: http://www.intel.com/products/processor/pentium4/index.htm
12. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Ann. 261, 513–534 (1982)
13. Nguyen, P.Q.: The dark side of the hidden number problem: Lattice attacks on DSA. In: Proc. of the Workshop on Cryptography and Computational Number Theory (CCNT '99), Basel, CH, pp. 321–330. Birkhäuser (2001)
14. Nguyen, P.Q., Shparlinski, I.: The insecurity of the digital signature algorithm with partially known nonces. J. Cryptology 15(3), 151–176 (2002)
15. Nguyen, P.Q., Stehlé, D.: Floating-point LLL revisited. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005)
16. Nguyen, P.Q., Stern, J.: The two faces of lattices in cryptology. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 146–180. Springer, Heidelberg (2001)
17. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: The case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006)

18. Percival, C.: Cache missing for fun and profit (2005), URL: http://www.daemonology.net/papers/htt.pdf
19. OpenBSD project members. OpenSSH Suite. URL: http://www.openssh.com/
20. Shoup, V.: NTL: A Library for doing Number Theory. URL: http://www.shoup.net/ntl/
21. Shparlinski, I., Winterhof, A.: A hidden number problem in small subgroups. Mathematics of Computation 74(252), 2073–2080 (2005)

Appendix

Lemma 3 (Short solutions). *Given prime N and $s \in \mathbb{Z}_N$, let $\rho_1, \ldots, \rho_l$ be uniformly and independently distributed on $\mathbb{Z}_N$. Then the probability that the congruence*

$$\sum_{j=1}^{l} \rho_j x_j \equiv s \ (\mathbf{mod}\, N) \tag{20}$$

has a "short" non-trivial solution $(t_1, \ldots, t_l) \in (\mathbb{Z}_N)^l$ satisfying $|t_j|_N < T_j$, $1 \le j \le l$ is lower than

$$\frac{2^l \prod_{j=1}^{l} T_j}{N}. \tag{21}$$

Proof. Let $\mathbf{t} = (t_1, \ldots, t_l)$ be a non-zero l-tuple in $(\mathbb{Z}_N)^l$ with $t_k \neq 0$. There exist exactly N^{l-1} l-tuples

$$(\rho_1, \ldots, \rho_l) = \left(\rho_1, \ldots, \rho_{k-1}, (t_k)^{-1} \left(s - \sum_{j=1, j\neq k}^{l} \rho_j t_j \right) \mathbf{mod}\, N, \rho_{k+1}, \ldots, \rho_l \right)$$

such that $\mathbf{t}$ is a solution of (20). Consequently, there exist no more than

$$N^{l-1} \left(\left(\prod_{j=1}^{l} (2T_j - 1) \right) - 1 \right) < N^{l-1} 2^l \prod_{j=1}^{l} T_j$$

l-tuples $(\rho_1, \ldots, \rho_l)$ such that "short" non-trivial solution of (20) exists. Since N^l is total number of all l-tuples on $\mathbb{Z}_N$, the lemma follows. □

Proof (of Lemma 1 on Short vectors in $\mathcal{L}$). Let $\boldsymbol{\Delta} \in \mathcal{L}$ be such that $\|\boldsymbol{\Delta}\|_\infty < \kappa_D \delta < 1$. Then

$$\left| e_i N + \alpha_i \sum_{j=1}^{d} 2^{\pi_j} y_j + \sum_{j=1}^{l_i} \rho_{i,j} t_{i,j} \right| = |\Delta_i| < 1, \quad 1 \le i \le d \tag{22}$$

$$\left| \frac{\delta}{2^{\nu_j}} y_j \right| = |\Delta_{d+j}| < \kappa_D \delta, \ 1 \le j \le m \tag{23}$$

$$\left| \frac{\delta}{2^{\mu_{i,j}}} t_{i,j} \right| = \left| \Delta_{d+m+j+\sum_{u=1}^{i-1} l_u} \right| < \kappa_D \delta, \quad \begin{matrix} 1 \le i \le d \\ 1 \le j \le l_i \end{matrix} \tag{24}$$

respectively implying

$$\left|\alpha_i \sum_{j=1}^{d} 2^{\pi_j} y_j + \sum_{j=1}^{l_i} \rho_{i,j} t_{i,j}\right|_N = 0, \qquad 1 \le i \le d \tag{25}$$

$$|y_j| < \kappa_D 2^{\nu_j}, \quad 1 \le j \le m \tag{26}$$

$$|t_{i,j}| < \kappa_D 2^{\mu_{i,j}}, \quad 1 \le i \le d, 1 \le j \le l_i, \tag{27}$$

since the expression on the left-hand side of (22) is an integer. Furthermore, (25) is equivalent to the congruence

$$\sum_{j=1}^{l_i} \rho_{i,j} t_{i,j} \equiv -\alpha_i \sum_{j=1}^{d} 2^{\pi_j} y_j \ (\mathbf{mod}\, N), \quad 1 \le i \le d. \tag{28}$$

To prove (i), we have to show the probability P_F that for all i, $1 \le i \le d$ there exists j, $1 \le j \le l_i$ such that $t_{i,j} \not\equiv 0 (\mathbf{mod}\, N)$ is bounded above as $P_F < \frac{(2\kappa_D)^{L+m} 2^{\tau+\xi}}{N^d}$. Regarding the algorithmic viewpoint of the whole paper, it is worth emphasizing the following probability elaboration is focused on the event that an "unwanted" vector does exist in the lattice at all, rather than investigating the properties of a particular vector chosen. The algorithmic interpretation is then that, obviously, the particular vector computed cannot have the properties that no such vector in the lattice $\mathcal{L}(\mathcal{I}, \delta)$ has.

Fix an m-tuple $(y_1, \ldots, y_m) \in \mathbb{Z}^m$ and define $R_i = -\alpha_i \sum_{j=1}^{d} 2^{\pi_j} y_j \,\mathbf{mod}\, N$. The substitution to congruence (28) gives

$$\sum_{j=1}^{l_i} \rho_{i,j} t_{i,j} \equiv R_i \ (\mathbf{mod}\, N), \quad 1 \le i \le d. \tag{29}$$

Lemma 3 states non-trivial solution of (29) satisfying the bounds (27) exists with the probability

$$p_i(y_1, \ldots, y_m) < \frac{2^{l_i} \prod_{j=1}^{l_i} \kappa_D 2^{\mu_{i,j}}}{N} = \frac{(2\kappa_D)^{l_i} 2^{\xi_i}}{N}. \tag{30}$$

For a fixed m-tuple $(y_1, \ldots, y_m)$, the probability that (29) and (27) can be non-trivially satisfied for all i, $1 \le i \le d$ is

$$p(y_1, \ldots, y_m) \le \prod_{i=1}^{d} p_i(y_1, \ldots, y_m) < \prod_{i=1}^{d} \frac{(2\kappa_D)^{l_i} 2^{\xi_i}}{N} = \frac{(2\kappa_D)^{L} 2^{\xi}}{N^d}. \tag{31}$$

There is no more than $\prod_{j=1}^{m} 2\kappa_D 2^{\nu_j} = (2\kappa_D)^m 2^\tau$ m-tuples $(y_1, \ldots, y_m)$ that satisfy (26), therefore

$$P_F \le \sum_{\substack{\mathbf{y} = (y_1, \ldots, y_m) \\ \mathbf{y} \text{ satisfies (26)}}} p(y_1, \ldots, y_m) < \frac{(2\kappa_D)^{L+m} 2^{\tau+\xi}}{N^d}. \tag{32}$$

To prove the congruence (ii) holds, it suffices to substitute $t_{w,j} \equiv 0 \ (\mathbf{mod}\, N)$, $1 \leq j \leq l_w$ from (i) to (28), i.e.

$$\sum_{j=1}^{m} 2^{\pi_j} y_j \equiv -(\alpha_w)^{-1} \sum_{j=1}^{l_w} \rho_{w,j} t_{w,j} \equiv 0 \ (\mathbf{mod}\, N), \tag{33}$$

since $(\alpha_w)^{-1} \,\mathbf{mod}\, N$ exists, because $\alpha_w \not\equiv 0 \,\mathbf{mod}\, N$ and N is a prime.

Finally, substituting (ii) to the congruence (28), i.e.

$$\sum_{j=1}^{l_i} \rho_{i,j} t_{i,j} \equiv -\alpha_i \sum_{j=1}^{m} 2^{\pi_j} y_j \equiv 0 \ (\mathbf{mod}\, N), \tag{34}$$

finishes the proof of (iii). □