Attacking DSA Under a Repeated Bits Assumption

P.J. Leadbitter, D. Page, and N.P. Smart

Dept. Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
{peterl,page,nigel}@cs.bris.ac.uk

Abstract. We discuss how to recover the private key for DSA style signature schemes if partial information about the ephemeral keys is revealed. The partial information we examine is of a second order nature that allows the attacker to know whether certain bits of the ephemeral key are equal, without actually knowing their values. Therefore, we extend the work of Howgrave-Graham, Smart, Nguyen and Shparlinski who, in contrast, examine the case where the attacker knows the actual value of such bits. We also discuss how such partial information leakage could occur in a real life scenario. Indeed, the type of leakage envisaged by our attack would appear to be feasible than that considered in the prior work.

1 Introduction

In previous work [4], Howgrave-Graham and Smart introduced a lattice based attack on DSA and EC-DSA in which they assumed that the adversary could obtain a certain number of bits of the secret ephemeral key for each message signed. By knowing only a few bits of each ephemeral key, an attacker could use their method to recover the entire secret and hence break the underlying cryptosystem. This method is related to the attacks of Bellare et. al. [1] and Bleichenbacher [2] who looked at issues related to poor generation of the random numbers which should be used in DSA/EC-DSA. Nguyen and Shparlinski [12, 13] subsequently extended the work of [4] to produce a more rigorous attack.

The concept of revealing secret information from a secure device such as a smart-card was made practical by the introduction of side-channel analysis. Specifically, on an undefended device, a simple power analysis (SPA) attack could leak a small number of bits from an exponentiation. When combined with the lattice techniques above, this leakage would result in the static private key being determined. However, defences against side-channel analysis are both increasingly well understood and frequently used in the field. Therefore the assumption that an attacker may be able to determine a specific set of bits from the ephemeral secrets is less probable than when the original attacks were first published. It is far more probable that second order, seemingly innocuous information can still be recovered and used by the attacker, even if defence against first order leakage is implemented.

Consider the possibility that an attacker can determine some relation amongst the bits of the secret ephemeral key rather than their specific values. For example, if the target device were using a window method for exponentiation, by examining the data transfers across the bus it could be possible for the attacker to determine that the first window of bits is equal to the second window of bits. This would be the case whenever the same value was fetched from memory in successive iterations of the main exponentiation loop. In such a situation we would have

$$k_i = z_i + 2^t y_i + 2^{t+w} y_i + 2^{t+2w} x_i$$

where x_i, y_i and z_i are variables satisfying

$$x_i < 2^{l-t-2w}, \quad y_i < 2^w, \quad z_i < 2^t,$$

for a secret of l bits in length. We use this information to formulate a lattice reduction problem which, when solved, gives us the value of the static secret key.

In this paper, for convenience, we investigate the case where $z_i = t = 0$ so k_i is the concatenation

$$y_i \| y_i \| x_i.$$

Such a scenario requires we take on average 2^w samples before one of this form should present itself by chance. This simplification is advantageous since it allows us to keep the lattice dimension small, speeding up the lattice reduction stage of the attack. A successful attack yields the private key of the signer enabling the attacker to impersonate his victim and forge a signature of any message without the victim's consent or knowledge of his having done so. Although we focus on the applicability to DSA/EC-DSA, we note that it is possible to apply the attack to protocols with a similar signing equation such as Schnorr signatures.

We organise our work as follows. To justify our assumption that obtaining relational information about bits in the ephemeral secret, in Section 2 we start by investigating situations where such leakage could occur. We then recap on the DSA/EC-DSA algorithm and basics of lattice basis reduction in Section 3. In Section 4 we examine how one embeds the information obtained from the side channel into a lattice before reporting on some experimental results from our technique in Section 5. Finally, we present some concluding remarks in Section 6.

2 Possible Attack Scenario

Side-channel analysis is a fairly new but increasingly effective cryptanalytic method that focuses on the implementation of an algorithm rather than the specification. By observing an implementation being executed, the attacker can make correlations between the events that occur in the host processor and the data being processed. Perhaps the most famous of these types of attack involve timing [7, 3] and power analysis [8]. In the former method, the attacker uses execution timings for either the whole algorithm or constituent parts to reason about the execution flow. For example, an implementation might take longer to run if a conditional branch is taken than if it is not taken. If that branch depends on a secret, key related value then being able to determine if it was taken or not can reveal said value. The later method uses the amount of power a processor uses to infer what operations and data items are being processed. A multiplication, for example, has a distinct profile within a power usage trace and will differ considerably from an addition. Furthermore, it is possible to discover the values of information being written into register or transfered across a bus since the state change in underlying logic will be different, and hence draw a different amount of power, depending on what values are used. If these values are assumed secret as part of the algorithm specification, the attacker is granted an easy and dangerous method of bypassing whatever hard mathematical problem the cryptosystem is based. There are two well accepted methods for performing power analysis attacks: simple power analysis (SPA) where only a single profile is enough to reveal information and differential power analysis (DPA) where correlation between multiple profiles is used to mount attacks that might otherwise fail in the SPA setting.

These techniques are made more dangerous by the environment in which they exist and the processing devices that are involved. Traditionally, side-channel attacks are mounted against accessible, portable processing units such as smartcards. Such devices are attractive to the attacker since they carry a potentially valuable, money or identity related payload and the physical access required for attacks is easier than in other cases. Furthermore, it has consistently been shown that a skilled engineer can mount side-channel attacks with low cost, commodity equipment essentially lowering the bar in terms of the investment required to break a given cryptosystem.

Often in side-channel attacks, directly revealing secret information is made difficult either by the inherent problems of mounting the profiling phase to collect observations of execution, or by defences employed in either hardware or software by the system designers. Such defences aim to reduce the amount of exploitable information that can be collected by the attacker. However, it has often been the case that seemingly innocuous information can still be harnessed by the attacker to their advantage thereby enabling new attacks that were not thought possible. Three such examples of new attack methods involve fixed table based implementations of elliptic curve (ECC) point multiplication [14, 18]; so called address-bit DPA which uses address calculation rather than data utilisation to provide information; and cache directed analysis of block ciphers [16, 17].

2.1 Table Based Exponentiation

Consider the following algorithm for computing an ECC point multiplication using the windowed, or w-ary, method.

- *Preprocessing Stage*

  - $T_i \leftarrow \mathcal{O}$.
  - For $i = 1$ upto $2^w - 1$.
    * $T_i \leftarrow T_{i-1} + P$.
- *Main Loop*
  - $Q \leftarrow \mathcal{O}$.
  - For $i = |k|/w$ downto 0.
    * For $j = 0$ upto w.
      · $Q \leftarrow 2Q$.
    * $Q \leftarrow Q + T_{k_i}$.
  - Return Q.

In order to improve upon standard methods, the algorithm precomputes a table containing small multiples of P. Using this table, we process the multiplier k in chunks of w bits in size rather than one bit at a time, by writing

$$k = \sum_{i=0}^{|k|/w} k_i 2^{iw}.$$

This acts to reduce the number of additive operations we perform and hence accelerate execution.

Although this method is attractive where memory for the precomputation can be spared, it is vulnerable to side-channel attack. If the attacker can isolate the portion of execution where the point T_{k_i} is read from memory he can compare this to known bit patterns, discover which table index is being read and thereby recover k. Even if the point in question can not be uniquely determined for some reason, equality or relations between two points, and hence values of k_i, may be established which at least lessen the workload of the attacker using further analytic methods. This vulnerability was thought to be such a problem that Möller [11], among others, formulated a defence whereby each point in the table was subject to some form of projective randomisation so that the points are no longer fixed and hence present an observer with no useful information.

2.2 Address-bit DPA

Table based point multiplication algorithms are also attackable via address-bit DPA [5] since if one can focus on and recover the index k_i that is calculated and used to perform the load of T_{k_i}, the multiplier k is insecure. Defences against this new form of attack have been proposed [6] that mix a random value r into the index so that the access is performed as $T_{k_i \oplus r}$. If r is refreshed before each application of the point multiplication algorithm, the table accesses are permuted to some extent meaning that simply recovering the address of a given access does not reveal the corresponding part of the secret multiplier. The problem with this approach is that relationships between the permuted indices will be retained so that if $k_i = k_j$ then after application of the proposed defence, it is still true that $k_i \oplus r = k_j \oplus r$. If the attacker can recover this relational information either directly or as part of some higher-order statistical approach [10], it could perhaps be used to break the defence.

2.3 Cache-based cryptanalysis

Using the behaviour of the bus is one way to snoop on how the table is accessed but since it is located in memory and potentially accessed through a cache, the data dependent behaviour of said cache could also yield information. This form of attack was successfully performed against DES [16, 17] whereby the attacks processed many plaintexts and collected those which took the longest to operate on. The attackers then made the correlation that longer execution time means more cache misses and that more cache misses meant a higher probability that two S-box entries were distinct. Hence, by using only conglomerate information about the total time of execution the attackers used the statistical bias in their collected sample to break the algorithm using further processing involving a workload of 2^{24} DES applications. This attack was performed on and against a desktop computer with a normal cache memory and minimal profiling tools. Clearly a similar principle applies in the point multiplication algorithm described above. Under the admittedly gross assumption that the cache is initially empty, the accesses to T_{k_i} will provoke a series of cache hits or misses depending on if the point in question has been loaded before or not. Using this information, relations about the values of k_i that provoked the table accesses can be recovered. Indeed, direct probing of the cache hits and misses might not even be required since statistical methods as described above could be used to guess the required value from a biased set of collected results.

2.4 Attack Summary

Clearly, as in most side-channel attack methods, the ability to perform a phase of profiling that yields the required information is vital to success. The rest of this paper assumes that an attacker can construct such a profiling phase and extract relational information as described. That is, we assume the attacker can recover w_i, a set of relations about w bit sized windows of k, with the following form

$$\begin{aligned} w_0 &= w_1 \\ w_1 &\neq w_2 \\ &\dots \end{aligned}$$

This example indicates that window zero is equal to window one which in turn is not equal to window two. If $w = 4$ and we count from the least significant bit, this means bits zero to three of k are equal to bits four to seven and so on. It is imperative to note that in each case we have no idea about the actual value of the bits involved, only relations between them.

Under this assumption, we focus on lattice based mathematical techniques that could be used to exploit such information should the attacker be able to recover it, using multiple runs of DSA/EC-DSA style signature schemes. Although we should consider the effect of an algorithm under attack within context, i.e. within a system with a composite defence against a number of attack avenues, our goal is to explore the effect of neglecting to secure this sort of presumed innocuous side-channel information. As such, this work provides three main contributions: a potential side-channel attack technique, a warning to those implementing systems that may fall under attack and an advancement in lattice based analytic methods. All are useful since it is clearly important to understand new vulnerabilities, even of a potential or theoretical nature, so that they can be solved before production systems are released into the market.

3 Notation: Signature Schemes and Lattices

In this section we introduce the notations and ideas required in subsequent discussion of our attack technique. In particular we recap on DSA style signature schemes and the notion of lattice basis reduction.

3.1 DSA-style Signature Schemes

The DSA algorithm, or equivalently EC-DSA, works in a finite abelian group G of prime order q generated by g. The private key is an integer $\alpha \in \{0, \ldots, q-1\}$, and the public key is the group element $y = g^{\alpha}$. We assume a conversion function

$$f : G \longrightarrow \mathbb{F}_q.$$

For DSA this function is given by

$$f : \begin{cases} G < \mathbb{F}_p^* & \longrightarrow \quad \mathbb{F}_q \\ \quad h & \longmapsto h \pmod q. \end{cases}$$

Whilst for ECDSA the conversion function is given by

$$f : \begin{cases} E(\mathbb{F}_p) & \longrightarrow \quad \mathbb{F}_q \\ \quad P & \longmapsto x(P) \pmod q, \end{cases}$$

where we interpret the x coordinates of P, denoted $x(P)$, as an integer before reduction modulo q.

Signing: To sign a message m, the owner of the key α selects an ephemeral secret k and computes

$$r = f(g^k)$$

before evaluating the signing equation

$$s = (H(m) + r\alpha)/k \pmod k.$$

The signature on the message m is then the pair (r, s).

Verification: To verify a signature (r, s) on a message m one first computes

$$a = H(m)/s \pmod q \text{ and } b = r/s \pmod q.$$

One then checks that

$$\begin{aligned} f\left(g^a y^b\right) &= f\left(g^{(H(m)+r\alpha)/s}\right) \\ &= f\left(g^{ks/s}\right) = f(g^k) \\ &= r. \end{aligned}$$

3.2 Lattice Basis Reduction

We first fix a positive integer d. For our purposes a lattice is a $\mathbb{Z}$-module spanned by n-linearly independent vectors in $\mathbb{R}^d$. The spanning set $\{b_1, \ldots, b_d\}$ is called the basis of the lattice. If we let the $d \times d$ matrix B be defined by column i being equal to lattice basis vector b_i then the associated lattice L is given by the set

$$L = \{B \cdot z : z \in \mathbb{Z}^d\}.$$

Lattice bases, and hence bases matrices, are unique up to multiplication on the right by an element of $GL_d(\mathbb{Z})$. Hence the integer

$$\Delta(L) = |\det(B)|$$

is well defined and does not depend on the actual basis being considered.

Amongst all possible basis there are some which are "better" than others, however finding a "good" basis and defining what one means by "good" can be quite difficult. In 1983 Lenstra, Lenstra and Lovász [9] defined a notion of a "good" lattice basis and gave a polynomial time algorithm to reduce an arbitrary lattice basis to one which satisfied their conditions. A basis which is reduced in the sense of Lenstra, Lenstra and Lovász is called LLL-reduced. We do not give the definition and algorithm here but simply refer the reader to [9] for more details. However, we do require the following result about LLL-reduced lattice bases

Theorem 1. *If $B = \{b_1, \ldots, b_d\}$ denotes an LLL-reduced basis for the lattice L then*

1. *For all $x \neq 0$ in the lattice L we have, for some constant c,*
$$\|b_1\|^2 \leq c\|x\|^2.$$
The constant c in the above statement can be taken to be 2^{d-1}.
2. *We have*
$$\|b_1\| \leq 2^{(d-1)/4} \Delta(L)^{1/d}.$$

The above theorem tells us that the first vector in an LLL-reduced basis is a close approximation to the smallest vector in a lattice and that the lattice size is approximately $\Delta(L)^{1/n}$. One should note that the problem of finding the smallest non-zero vector in a lattice appears to be a very hard computational problem, but that the LLL-algorithm provides an approximation in polynomial time.

4 Embedding into a Lattice Problem

Suppose we run DSA/EC-DSA repeatedly and, through the side-channel attacks mentioned previously, or otherwise, we find $n+1$ signatures where the ephemeral key k_i is of the form, for $i = 0, \ldots, n$,

$$y_i \| y_i \| x_i$$

where

$$q \approx 2^l, \; y_i < 2^w \text{ and } x_i < 2^{l-2w},$$

i.e. we have

$$k_i = x_i 2^{l-2w} + y_i(1 + 2^w)$$

where x_i and y_i are unknowns. Note that it will take on average $n2^w$ signatures to obtain all this data if ephemeral keys are chosen at random and the means of detecting whether such an ephemeral key occurs is one hundred percent accurate. From the $n+1$ signing equations

$$s_i = (H(m_i) + r_i \alpha) k_i^{-1} \pmod q \text{ for } i = 0, \ldots, n,$$

we can form n equations

$$r_0 s_i k_i - r_i s_0 k_0 = r_0 H(m_i) - r_i H(m_0) \pmod q \text{ for } i = 1, \ldots, n,$$

by eliminating the variable α corresponding to the static private key. Substituting $k_i = x_i 2^{l-2w} + y_i(1 + 2^w)$ we have,

$$y_i = a_i + b_i x_0 + c_i x_i + d_i y_0 + \lambda_i q \text{ for } i = 1, \ldots, n,$$

for some $\lambda_i \in \mathbb{Z}$ where

$$\begin{aligned}
a_i &= (2^w + 1)^{-1} s_i^{-1} (H(m_i) - H(m_0) r_i r_0^{-1}) \\
b_i &= 2^{l-2w} (2^w + 1)^{-1} s_i^{-1} s_0 r_i r_0^{-1} \\
c_i &= -2^{l-2w} (2^w + 1)^{-1} \\
d_i &= s_i^{-1} s_0 r_i r_0^{-1}
\end{aligned}$$

Embedding these equations into the $d = 2n + 3$ dimensional lattice L generated by the columns of the matrix

$$E = \left(\begin{array}{c|c|ccc|c|ccc}
\beta & 0 & & 0 & & 0 & & 0 & \\
\hline
0 & \gamma & & 0 & & 0 & & 0 & \\
\hline
0 & 0 & \gamma & & 0 & 0 & & & \\
\vdots & \vdots & & \ddots & & \vdots & & 0 & \\
0 & 0 & 0 & & \gamma & 0 & & & \\
\hline
0 & 0 & 0 & \ldots & 0 & \delta & 0 & \ldots & 0 \\
\hline
a_1 & b_1 & c & & 0 & d_1 & \delta q & & 0 \\
\vdots & \vdots & & \ddots & & \vdots & & \ddots & \\
a_n & b_n & 0 & & c & d_n & 0 & & \delta q
\end{array} \right).$$

Then we have that $E \cdot \underline{x} = \underline{z}$ where

$$\underline{x}^t = (1, x_0, x_1, \ldots, x_n, y_0, \lambda_1, \ldots, \lambda_n),$$
$$\underline{z}^t = (\beta, \gamma x_0, \gamma x_1, \ldots, \gamma x_n, \delta y_0, \delta y_1, \ldots, \delta y_n).$$

In addition we would like the target vector $\underline{z}$ to be a short vector in the lattice. Hence, we need to choose the weights β, γ and δ in such a way as to increase the likelihood that $\underline{z}$ is indeed a short vector and hence likely to be found by lattice basis reduction. In our implementation we chose β, γ and δ to be related by $\gamma = 2^{2w-l}\beta$ and $\delta = 2^{-w}\beta$, to see why this is a good choice we need perform the following calculation.

From Theorem 1, a useful heuristic for predicting the success of such a lattice attack is to check whether our target vector z satisfies

$$\|z\| \leq \Delta(L)^{1/d}.$$

It is easy to see that, for our t

$$\begin{aligned} \Delta(L)^2 &= \beta\gamma^{n+1}\delta^{n+1}q^n \\ &= \beta^{2n+3}2^{(n+1)((2w-l)-w)}2^{ln} \\ &= \beta^{2n+3}2^{(n+1)(w-l)+ln} \\ &= \beta^{2n+3}2^{nw+w-l}. \end{aligned}$$

and

$$\begin{aligned} \|z\|^2 &= \beta^2 + \sum_{i=0}^{n} \left(\gamma^2 x_i^2 + \delta^2 y_i^2\right) \\ &\leq \beta^2 \left(1 + (n+1)2^{4w-2l}2^{2l-4w} + (n+1)2^{-2w}2^{2w}\right) \\ &= \beta^2(2n+3). \end{aligned}$$

Hence, for our lattice based approach to have a chance of succeeding, we must have

$$\sqrt{2n+3} \leq 2^{(nw+w-l)/(2n+3)}.$$

In practice l is 160 and if d is much larger than 300, the computation of LLL reduced bases takes a prohibitively long time. If we assume reduction of 300 dimension lattices is both feasible and results in vectors close to the minimum (which is a very optimistic assumption), we are assuming that $n \approx 100$. We will recover the full secret if

$$3.83 \approx \log_2\left(\sqrt{2n+3}\right) \leq (101w - 160)/203.$$

i.e.

$$w \geq (3.83 \cdot 203 + 160)/101 = 9.28$$

Thus we expect 10 equal bits in consecutive positions to be sufficient in our problem.

5 Experimental Results

In order to get an idea of how successful this sort of attack might be, we ran a large number of experiments. Our initial goal was to sweep a reasonable area of the parameter space and determine a rough success rates for different combinations of window size and number of messages. However, as the number of messages grows the lattice dimensions and hence the time take to perform the attack also grows. This effect means that a great deal of processing time is required to perform attacks with a large number of messages. To enable completion of our experiments within a reasonable time frame, we distributed the workload across a network of around fifty Linux workstations each incorporating 2 GHz Intel Pentium 4 processors and around 512 Mb of memory. Using these machines we conducted one hundred runs of each combination of parameters and quote the success rate of these attacks in Table 1.

	Messages					
Window	10	20	30	40	50	60
5	0%	0%	0%	0%	0%	0%
6	0%	0%	0%	0%	0%	0%
7	0%	0%	0%	0%	0%	0%
8	0%	0%	0%	0%	0%	0%
9	0%	0%	0%	12%	26%	30%
10	0%	0%	41%	96%	99%	98%
11	0%	0%	100%	100%	100%	100%
12	0%	31%	100%	100%	100%	100%
13	0%	99%	100%	100%	100%	100%
14	0%	100%	100%	100%	100%	100%
Time	0.38	4.72	21.70	106.28	317.21	570.89

Table 1. A table showing the success rate of attacking a 160 bit exponent with variable window size and different numbers of messages. Note that window sizes below 9 and number of messages below 20 yielded no successful attacks. Also note that the number of messages is the dominant factor in how long each attack takes and that we measure the average time taken in minutes.

Each attack involved two successive LLL reductions. The first LLL application used a floating point implementation of the Schnorr–Euchner algorithm [15] using deep insertions. Due to floating point errors this only provided an approximation to an LLL reduced basis. To obtain a fully reduced basis the version of De Weger [19] was then applied to the output basis from the Schnorr–Euchner algorithm.

There are several interesting features in these results. Firstly, it is clear that window sizes below 9 and number of messages less than 20 yielded no successful attacks. In terms of window size this is unfortunate since we would expect real attack scenarios to utilise small windows, for example window widths of size 4 or 5, that correspond to table indices or addresses for example. Secondly, there is a fairly polar behaviour in terms of success in that an attack seems to either work either nearly all of the time or nearly none of the time. Again, this is unfortunate from a practical stand point since as an attacker we can tolerate probabilistic success if the parameters allow more realistic choices.

The polar behaviour is a common feature of LLL experiments. For each lattice Λ, we can consider the value

$$D(\Lambda, a) = |\{v \in \Lambda : 0 < \|v\| < a\}|$$

We shall call function D the *norm distribution* of Λ. In out attack, we looked at lattices of a particular form

$$L(n, \beta, \gamma, \delta, \boldsymbol{a}, \boldsymbol{b}, c, \boldsymbol{d})$$

in which we know the size of one of the non-zero lattice points is small; our target vector $\boldsymbol{z}$ is less than some number Z. For fixed w and n, our norm distribution $D(L, \cdot)$ changes very little from experiment to experiment. When $D(L, Z+\epsilon) = 1$, where ϵ is a small number that accounts for the LLL error as an SVP oracle, we expect the attack to succeed. Moreover we expect it to succeed for all the other experiments of the same w and n values. Similarly when $D(L, Z+\epsilon)$ is large, we expect failure every time. Probabilistic success occurs when $D(L, Z+\epsilon)$ is small but larger than 1. Compared to the huge number of lattice points we are dealing with dwarfs the number of experiments we were able to do, we see probabilistic success on only a few of the parameter choices.

Our results only seem to succeed for $n \leq 9$. We believe this to be the limit of attacks using this style of lattice. A different lattice style could have quite a different norm distribution and could respond better to LLL, reducing our ϵ error term. This could yield much more favourable results than those presented here and remains an open problem. Indeed in the DSA attacks with several known bits, the success rate has been raised by simply inputting the information in a different way, see [12] and [13].

To get a better idea of how the attack behaves when using parameters that are ideal from an attackers point of view, we started a second set of experiments that focus on a window size of four but with much larger number of messages. We expected this to be more suitable in practice since, as discussed in Section 2, four bit indices are often used in table driven exponentiation. If capturing relations between these indices is possible, we would therefore be interested in knowing their potential for use in an attack. Unfortunately, the results of these experiments were inconclusive due to the length of time and amount of memory required to complete each one. The bottleneck proved to be the efficiency of our LLL implementation which, with a large number of messages, required so much memory to store the lattice that the virtual memory system was unable to maintain an acceptable performance level. Although negative, this second result does provide us some information in the context of our investigation. That is, forcing an attacker to collect many signatures is clearly a good way to foil attack in a practical situation since performing the lattice reduction is too computationally hard.

6 Conclusions

We have presented an interesting extension of prior work on lattice reduction used in the context of side-channel attacks. We weaken the assumptions of previous work so that it is more probable that the profiling phase of an attack will recover useful information, even when defence measures are deployed against other techniques. By extending prior work that assumes an attack can obtain the value of secret information by allowing them simply to uncover relationships between different parts of said information. This is especially dangerous in the context of signature schemes such as DSA/EC-DSA where such leakage can totally reveal the underlying secret.

However, the results from our experimentation are not as positive as the initial attack scenario. We found that the attacker would need to collect relationships about a large number of bits in contrast with knowing the value of a small number of bits in previous work. Such large relationships would be difficult to collect with existing side-channel analytic techniques and, in this respect, further work is needed to extend the attack. We expect that continued research into physically obtaining bit relationships from a target device and more efficient implementations of the lattice reduction stage might make our attacker more feasible in the future.

References

1. M. Bellare and S. Goldwasser and D. Micciancio. "Pseudo-Random" Number Generation Within Cryptographic Algorithms: The DSS Case. In *Advances in Cryptology – EUROCRYPT '97*, Springer-Verlag LKNCS 1233, 277–291, 1997.
2. D. Bleichenbacher. On the generation of DSS one-time keys. Preprint, 2001.
3. D. Boneh and D. Brumley. Remote Timing Attacks Are Practical. To appear in *12th USENIX Security Symposium*, USENIX Press, 2003.
4. N. Howgrave-Graham and N.P. Smart. Lattice attacks on digital signature schemes. *Designs, Codes and Cryptography*, **23**, 283–290, 2001.
5. K. Itoh, T. Izu and M. Takenaka. Address-Bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Springer-Verlag LNCS 2523, 129–143, 2002.
6. K. Itoh, T. Izu and M. Takenaka. A Practical Countermeasure Against Address-Bit Differential Power Analysis. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Springer-Verlag LNCS 2779, 382–396, 2003.
7. P.C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology – CRYPTO '96*, Springer-Verlag LNCS 1109, 104–113, 1996.
8. P.C. Kocher, J. Jaffe and B. Jun. Differential Power Analysis. In *Advances in Cryptology – CRYPTO '99*, Springer-Verlag LNCS 2139, 388–397, 1999.

9. A.K. Lenstra, H.W. Lenstra and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, **261**, 515–534, 1982.
10. T.S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Springer-Verlag LNCS 1965, 238–251, 2000.
11. B. Möller. Parallelizable Elliptic Curve Point Multiplication Method with Resistance against Side-Channel Attacks. In *Information Security (ISC)*, Springer-Verlag LNCS 2433, 402–413, 2002.
12. P.Q. Nguyen and I.E. Shparlinski. The insecurity of the Digital Signature Algorithm with partially known nonces. *J. Cryptology*, **15**, 151–176, 2002.
13. P.Q. Nguyen and I.E. Shparlinski. On the insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Designs, Codes and Cryptography*, To appear.
14. W. Schindler. A Combined Timing and Power Attack. In *5th Workshop on Practice and Theory in Public Key Cryptosystems (PKC)*, Springer-Verlag LNCS 2274, 263–279, 2002.
15. C.P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. In *Proc. FCT 1991*, Springer-Verlag LNCS 529, 68–85, 1991.
16. Y. Tsunoo, E. Tsujihara, K. Minematsu and H. Miyauchi. Cryptanalysis of Block Ciphers Implemented on Computers with Cache. In *International Symposium on Information Theory and Its Applications (ISITA)*, 2002.
17. Y. Tsunoo, T. Saito, T. Suzaki, M. Shigeri and H. Miyauchi. Cryptanalysis of DES Implemented on Computers with Cache. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Springer-Verlag LNCS 2779, 62–76, 2003.
18. C.D. Walter and S. Thompson. Distinguishing Exponent Digits by Observing Modular Subtractions. In *Topics in Cryptology (CT-RSA)*, Springer-Verlag LNCS 2020, 192–207, 2001.
19. B.M.M. de Weger. Solving exponential diophantine equations using lattice basis reduction. *J. Number Theory*, **26**, 325–367, 1987.