Weaknesses in Current RSA Signature Schemes

Juliane Krämer Dmitry Nedospasov Jean-Pierre Seifert
{juliane,dmitry,jpseifert}@sec.t-labs.tu-berlin.de

Security in Telecommunications
Technische Universität Berlin and Deutsche Telekom Innovation Laboratories
Germany

Abstract. This work presents several classes of messages that lead to data leakage during modular exponentiation. Such messages allow for the recovery of the entire secret exponent with a single power measurement. We show that padding schemes as defined by industry standards such as PKCS#1 and ANSI x9.31 are vulnerable to side-channel attacks since they meet the characteristics defined by our classes. Though PKCS#1 states that there are no known attacks against RSASSA-PKCS1-v1_5, the EMSA-PKCS1-v1_5 encoding in fact makes the scheme vulnerable to side-channel analysis. These attacks were validated against a real-world smartcard system, the Infineon SLE78, which ran our proof of concept implementation. Additionally, we introduce methods for the elegant recovery of the full RSA private key from blinded RSA CRT exponents.

Keywords: RSA, PKCS#1, ANSI x9.31, Side-Channel Attacks, Simple Power Analysis, CRT, Exponent Blinding

1 Introduction

Side-channel attacks exploit information leaked by the physical characteristics of a cryptosystem [8, 9, 17]. A common side-channel attack is power analysis. Power analysis can be categorized into two subcategories, *simple power analysis* (*SPA*, methods requiring few measurements) and *differential power analysis* (*DPA*, methods requiring many measurements) [12]. Since it is commonly impossible to recover the data being leaked in a single measurement, adversaries are often forced to perform DPA to recover data in its entirety. In turn, countermeasures, e.g., blindings, are implemented to counteract the attacks and to thwart DPA.

In this work, we attack the RSA signature process by performing simple power analysis to recover the potentially blinded secret exponent. We present several classes of messages that lead to data leakage during modular exponentiation. Specifically, we show that several properly formatted standardized input messages, including the message encodings of PKCS#1 [18] and ANSI x9.31 [1], meet the criteria defined by these classes. Thus, we show that compliance with industry standards can in fact lead to data leakage, although these standards are considered to be secure message encodings.

The analysis was performed against our proof of concept (POC) implementation running on an Infineon smartcard system, which performed ZDN-based modular multiplication [7]. This setup allowed us to test all classes of input messages presented in this work. For all classes of input messages, the SPA yielded the entire private exponent or the entire blinded private exponent that was used for the signature process.

Recovering a potentially blinded exponent is sufficient to sign messages in the RSA signature scheme. However, this is not true for RSA CRT. In the case of RSA CRT, the attacker must instead recover the full private key. Methods for recovering the full RSA private key have been known since 1978, whereas we present two specific methods for RSA CRT, with and without exponent blinding. Since several methods for recovering a secret exponent fail to recover all of the exponent bits, we present the approach *exponent un-blinding*, which enables an attacker to compute the full private key more efficiently in such cases. This method can cope with more noise, more efficiently than other known methods.

The main contributions of this paper are:

1. Categorization of common vulnerable message classes and the corresponding attack scenarios.
2. Demonstrating that constant padding makes RSA signature schemes such as RSASSA-PKCS1-v1_5 and ANSI x9.31 vulnerable to side-channel analysis.
3. Practical validation of the attacks and the proposed attack scenarios against a proof of concept implementation on an Infineon smartcard system.
4. More efficient methods for recovering the full RSA CRT private key from blinded private exponents.

The paper is organized as follows: Section 2 presents necessary background information. In Section 3 we present several common types of input messages and explain attack vectors and scenarios which arise from certain characteristics of these messages. We categorize these characteristics into classes of messages which lead to data leakage. Any valid PKCS#1 and ANSI x9.31 message meets the criteria defined in one of these classes. We then demonstrate these attacks against an Infineon smartcard system running our POC implementation in Section 4. In Section 5, novel methods for the recovery of the full private key are explained for RSA CRT. Finally, we summarize the implications of our research in Section 6.

2 Background

In this section, we first give a brief introduction of the RSA and the RSA CRT signature scheme. We then explain the square-and-multiply algorithm for modular exponentiation and the ZDN algorithm for modular multiplication. Finally, we explain blinding techniques, which are used to thwart statistical attacks.

2.1 RSA CRT

Let (N, e) be the public RSA modulus and exponent, and $(p, q, d, \varphi(N))$ be the private key, satisfying $N = pq$ and $ed \equiv 1 \bmod \varphi(N)$. As the modulus N is

the product of two different primes, the *Chinese Remainder Theorem* (CRT) can be used to speed up the time intensive process of message signing by a factor of four [13, 16]. Instead of computing the RSA signature $s = m^d \bmod N$ with an exponent of the order of $n = \log_2(N)$ bits (assuming a small e), two modular exponentiations with $n/2$-bit exponents are performed. In this setting and without loss of generality $q < p$, we precompute $d_p = d \bmod (p-1)$, $d_q = d \bmod (q-1)$ and $q_{inv} = q^{-1} \bmod p$. These constants are also part of the private key [18]. They are used for the computations

$$s_p = m^{d_p} \bmod p \qquad \text{and} \qquad s_q = m^{d_q} \bmod q. \tag{1}$$

Subsequently, Garner's algorithm is used to yield the signature s of m:

$$s = s_q + (q_{inv} \cdot (s_p - s_q) \bmod p) \cdot q. \tag{2}$$

2.2 Square-and-Multiply for Modular Exponentiation

A commonly used algorithm for modular exponentiation is the modular *square-and-multiply* algorithm, which exploits the binary representation of the exponent, see Figure 1. The input for this algorithm is $(m,\ d,\ N)$ and its output, $s = m^d \bmod N$, is the signature of m. Let d_i, $i \in \{0, \ldots, l-1\}$, denote the i^{th} bit of d, i.e., d_0 is the least significant bit. Thus, l is the bit length of d and we have $l = \lfloor \log_2(d) \rfloor + 1$. Performing a modular exponentiation with this algorithm needs $\mathcal{O}(\log_2(d))$ operations, i.e., it has logarithmic complexity.

2.3 The ZDN Algorithm for Modular Multiplication

Whithin the square-and-multiply algorithm, modular multiplications are perfomed. *ZDN*-based modular multiplications consist of three major parts, computation of the "look-ahead" multiplication (`LABooth`) [5, 21], computation of the "look-ahead" reduction (`LARed`) [5, 21], and a subsequent 3-operand addition, which finally yields the resulting partial product, see Figure 2. `LABooth` is optimized to shift across constant bit strings, whereas `LARed` requires only several significant bits to compute the reduction. The look-ahead reduction is designed so that its average reduction is approximately the same as the one of the look-ahead multiplication.[1] Thanks to this high level of optimization, the three parts are executed in parallel and require just a single clock cycle [5]. The algorithm ensures that the intermediate result Z fulfills $|Z| \leq \frac{1}{3}N$, hence the name (two-thirds N is *zwei Drittel N* in German).

2.4 Blinding Techniques to Thwart Statistical Attacks

Both RSA and RSA CRT are vulnerable to differential side-channel attacks [8, 9, 17]. To prevent these statistical side-channel attacks, randomized *blinding* is used

[1] Both look-ahead sub-operations are explained in detail in [20, 21].

```
1  input: m, d, N
2  output: m^d mod N
3  k := log2(d) - 1, t := 1
4  while k >= 0
5    // square
6    // ZDN Mod-Mult
7    t = (t·t) mod N
8    if d_k = 1 then
9      // multiply
10     // ZDN Mod-Mult
11     t = (t·m) mod N
12   k := k - 1
13 endwhile
14 return t
```

Fig. 1. Modular Square-and-Multiply

```
1  input: t, m, N
2  output: m·t mod N
3  Z := 0, C := m
4  l := log2(t) + 1, c := 0
5  while l > 0 or c > 0 do
6    LABooth(t, &l, &s_t, &v_C)
7    LARed(Z, N, c, &s_z, &v_N)
8    s_C := s_Z - s_t
9    C := C·2^{s_C}
10   Z := Z·2^{s_Z} + v_C·C + v_N·N
11   c := c - s_C
12 endwhile
13 if Z < 0 then Z := Z + N
14 return Z
```

Fig. 2. ZDN-Based Modular Multiplication

to disguise intermediate results and to decouple the leaked information from the processed data. We explain three different blinding techniques of which two are vulnerable to our attack. In all cases, the integers r and r_1, r_2, respectively, are random λ-bit numbers, commonly $\lambda = 32$ [22]. A new r is chosen independently for every operation.

The first of these blinding techniques is called *exponent blinding* [22]. The blinded exponent is $d' := d + r \cdot \varphi(N)$. Due to Euler's theorem, the following equation holds true: $s = m^d \bmod N = m^{d+r\cdot\varphi(N)} \bmod N$.[2] The same blinding can be applied to both exponents when using RSA CRT. In this case, the blinded exponents are $d'_p = d_p + r_1 \cdot (p-1)$ and $d'_q = d_q + r_2 \cdot (q-1)$, respectively.

The second blinding technique is called *base blinding* [9] or *message blinding*. Base blinding decouples the side channel leakage from the input m. For a random λ-bit integer r its inverse modulo N is calculated, i.e., $r \cdot r^{-1} \equiv 1 \bmod N$. The blinded message is $m' := r^e \cdot m$. Instead of m, the blinded message is signed, yielding the blinded signature s'. The blinding is reversed by computing $s = (r^{-1} \cdot s') \bmod N$. Since this form of base blinding includes a computationally expensive inverse calculation in ${\mathbb{Z}_N}^*$, it is relatively unattractive for embedded systems. Alternatively, another form of base blinding can be used. Given two random λ-bit integers r_1, r_2 where $r_1 < r_2$, the exponentiation can be computed as follows, $s = \left[(r_1 \cdot N + m)^d \bmod (r_2 \cdot N)\right] \bmod N$. This form of base blinding is far less computationally expensive.

[2] It is very unlikely that the necessary condition $\gcd(m, N) = 1$ is not fulfilled.

3 SPA-Based Secret Exponent Recovery

In this section, we present several side-channel attacks which obtain the exponent of a modular exponentiation with a single-trace analysis. Most known methods for recovering exponents rely on statistical analysis [4, 8, 9, 19, 23]. As such, these methods require multiple exponentiations with the same exponent, and thus, can be prevented by exponent blinding. Instead, we consider approaches that are able to recover the entire exponent from a single modular exponentiation operation. These approaches also work whenever exponent blinding is used. First, we recall several known methods and then we present a list of criteria for the message that, when met, allow us to recover an exponent in a single trace.

3.1 Known Methods

In 2005, an attack that uses the specific input $m = -1$, i.e., $N - 1 \bmod N$, was presented [24]. This attack exploits the fact that whenever square-and-multiply is used, there are just three distinct pairs of operations, which are performed during exponentiation [14]. Due to the special input message, these distinct pairs result in three distinct power dissipation states, which can be identified within the power trace. Therefore, the bit pattern of the private exponent can be obtained by performing a single-trace SPA [14, 24]. The same approach was extended for RSA CRT.

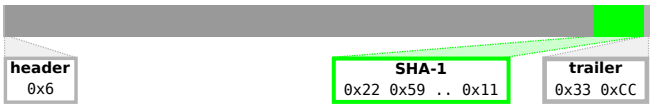
In 2010, an additional method for recovering secret data via a single-trace SPA emerged [3]. The authors consider systems, which utilize t-bit multipliers for performing long integer arithmetic, i.e., $t = 32$ or $t = 64$. If one or more t-bit strings of a message are equal to 0, i.e., 0-strings, the message will lead to data leakage in the power trace. The authors also describe several possible messages, which lead to data leakage, such as messages with a low Hamming weight, i.e., $m = 2^x$ where $x \leq \log_2(N)$. The authors mention that multiple constant t-bit strings, or constant strings that are longer in length, only increase the leakage even more. We demonstrate in this work that certain aspects of this attack are also applicable to systems that do not use t-bit multipliers of a certain length t, and consider systems that perform full length integer multiplication directly.

3.2 Classes of Input Messages

We present several classes of input messages and corresponding attack scenarios that lead to differences in the power consumption depending on the value of the exponent bit. As a result, whenever a cryptosystem performs a modular exponentiation with a message from one of these classes, the exponent bit sequence can be recovered. We validate these claims by performing an SPA against a POC implementation in Section 4.

Our attacks are based on messages that have constant bit strings, which can lead to data leakage. Specifically, we focus on two message types. The first class includes standardized messages, which consist largely of constant padding. In such cases, the constant padding of the leading bits constitutes a Leading

(a) 2048-bit PKCS#1 encoded message



(b) 2048-bit ANSI x9.31 encoded message

Fig. 3. Figures 3(a) and 3(b) are drawn to scale. The dark gray area of Figure 3(a) corresponds to the "heavier" Hamming weight of the leading `0xFF` of the RSASSA-PKCS1-v1_5 padding and the lighter gray of Figure 3(b) to the "lighter" Hamming weight of the leading `0xBB` padding of ANSI x9.31.

Constant Bit String (LCBS). In the second class of message we consider regions of the message, which are set or affected by user input. Usually this is a region of the least significant bits or trailing bits. Hence we refer to this class of message as Trailing Constant Bit String (TCBS).

The message classes described in this section allow us to distinguish between square and multiply operations. Once we are able to distinguish a square from a multiply, the bit pattern of the exponent can be recovered from a single power trace, as is demonstrated in Section 4.

Leading Constant Bit String: The first class of message we consider, is the *Leading Constant Bit String* (LCBS). These are messages in which the most significant bits consist mostly of constant 0- or 1-strings. LCBS messages are particularly interesting because many valid messages utilizing non-random padding schemes constitute LCBS messages. For example, we classify both RSASSA-PKCS1-v1_5 of PKCS#1 [18] because of the leading `0xFF` (11111111_2), and to a lesser extent ANSI x9.31 [1] with leading `0xBB`(10111011_2) as LCBS messages. Thus, attacks that utilize LCBS messages are potentially harder to defend against because such attacks do not necessarily rely on the chosen message attack model. In such a scenario, the data is leaked by **any** valid message. Though PKCS#1 states that there are no known attacks against RSASSA-PKCS1-v1_5 [18], we demonstrate that the EMSA-PKCS1-v1_5 encoding in fact makes the scheme vulnerable to side-channel analysis, see Section 4. In the case of the exemplary 2048-bit PKCS#1 message, over 84% of the message is padding, see Figure 3(a). In the case of the exemplary 2048-bit ANSI x9.31 message, over 91% of the message is padding, see Figure 3(b).

LCBS messages do not necessarily reduce the workload of the modular reduction on systems that do not perform multiplication and reduction in parallel. However, the consistent structure of the leading 0- or 1-strings ensures a reduced workload on highly optimized systems implementing algorithms like ZDN [5].

Trailing Constant Bit String: The second class of message we consider is the *Trailing Constant Bit String* (TCBS). This is an important classification because many cryptographic schemes operate on messages that contain only a relatively small variable region set or affected by user input. In most cases, this region is a relatively small portion of the least significant bits and the bulk of the message consists of padding. However, if an attacker is able to set as little as 5%-10% of the least significant bits by, for example, supplying the specified hash to the signature scheme directly, then the attacker would be able to recover the secret data independent of the padding scheme being used. In this scenario, even if randomized padding is used, a very small region of trailing bits is sufficient to leak the entire secret data. Note, standards such as PKCS#1 also define multiple hash algorithms that can be used. Potentially, an attacker could even increase the region affected by user input to be as large as 512 bits if he is allowed to provide, for example, SHA-512-based messages instead.

As with LCBS, TCBS messages can be as long as the modulus in bits and, as such, TCBS messages do not necessarily reduce the workload of the modular reduction. This is generally the case whenever multiplication and reduction are not computed in parallel. However, on highly optimized systems, i.e. those which implement ZDN [5], the constant trailing 0- or 1-strings ensure a reduced workload.

Short Messages: Though of little interest if the implementation enforces padding, the third class of messages we consider is the *short message*. Short messages are messages $m \ll N$, where N is the modulus of the modular exponentiation operation in question. Short messages can be considered LCBS messages with leading 0-strings. We consider short messages in this work, primarily because they exploit both the multiplication and reduction step of modular multiplication and achieve the greatest difference in the power consumption of squares and multiplies, respectively. This was also validated against our POC implementation, where padding checks could be disabled, see Section 4. Note that efficient implementations generally ignore, or shift across any leading 0-strings, which in conjunction with the relatively low Hamming weight of the entire message greatly reduces the workload of the multiplication step. Additionally, because of the short length in bits of the message m, the intermediate result of the multiplication step of the square-and-multiply algorithm increases by only $\log_2(m)$ bits in length prior to reduction. In comparison, during the square operation, the bit length of the intermediate result approximately doubles. As a result, in addition to the lower computational workload of the multiplication, such messages also reduce the computational workload of the modular reduction after a multiplication and potentially eliminate reduction completely, further lowering the power consumption of the multiply operation.

4 Proof of Concept

In this section we present the practical evaluation of the classes of messages described in Section 3 on a real-world system.

The cryptosystem analyzed in this work is an Infineon SLE78-based [7] smartcard system. The SLE78 features a cryptographic coprocessor known as the Crypto@2304T, which provides 2304-bit registers and ZDN-based modular multiplication [5]. In contrast to previous works such as [3], which focus on cryptosystems that use "short" bit length multipliers (i.e. 32 or 64-bit multipliers), the SLE78 performs full-length arithmetic operations by utilizing registers and logic capable of 1024-bit and 2048-bit modular multiplication. With ZDN-based modular multiplication, multiplication and reduction are computed in parallel in multiple iterations of the modular multiplication loop, see Section 2.3. This improves performance and further reduces register length requirements by ensuring $|Z| \leq \frac{1}{3}N$ for the partial product Z and the RSA modulus N.

When used in conjunction with algorithms like square-and-multiply, the highly optimized nature of this modular multiplication introduces additional weaknesses. The two important characteristics of ZDN-based modular multiplication, which were exploited in this work are:

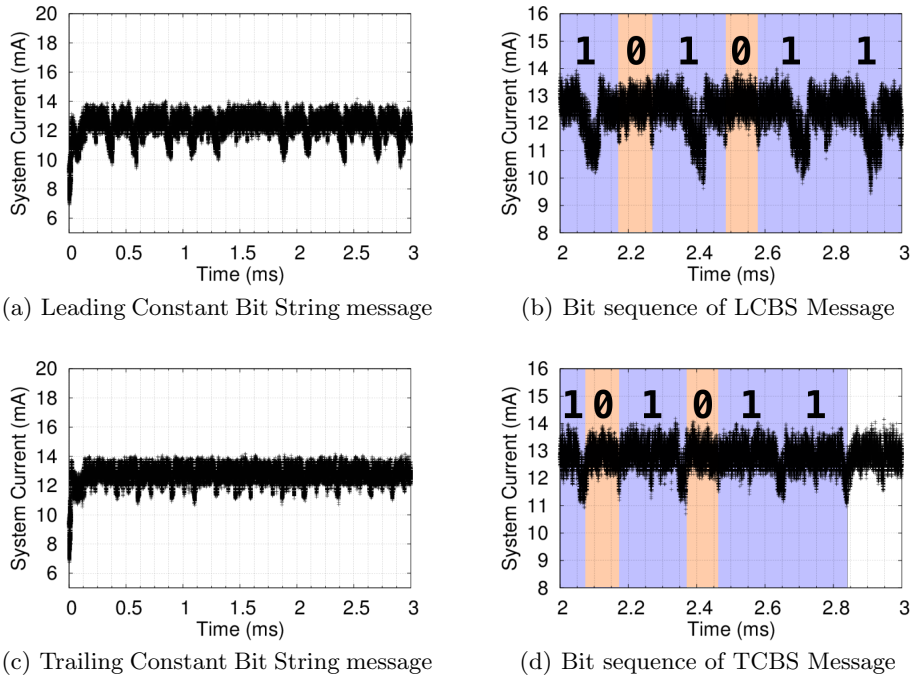
- `LABooth` ensures that the cryptosystem can shift across 0- (i.e., 00..00) and 1-strings (11..11) as well as 0- and 1-strings with isolated 1's and 0's, respectively, i.e., (0..010..0) and (1..101..1).
- `LARed` ensures that partial products are only actually reduced whenever they become too large, i.e. $|Z| > \frac{1}{3}N$.

By selecting messages, which meet the criteria outlined in Section 3, we exploit all of these characteristics of the algorithm. However, exploiting even any one characteristic of the algorithm allows for the recovery of the sequence of square and multiply operations, and thus, for the recovery of the secret exponent.

The system ran a proof of concept software implementation, which performed RSA signing. This implementation used square-and-multiply for modular exponentiation, the ZDN algorithm for modular multiplication, and it performed exponent blinding, as described in Section 2.2, 2.3 and 2.4. The system did not enforce padding, which allowed us to test all the message types described in Section 3, including short messages. The system was connected to a PC, which ran the client software, via a standard USB smartcard reader. The client software allowed us to select input messages and enable or disable additional software and hardware countermeasures.

Figures 4(a) and 4(c) show the first 3ms of the computation for a common exponent, but with the different classes of messages introduced in Section 3 as the input. The modulus of the modular exponentiation was 1024 bits in length for all the input message classes. For comparison since truly random messages do not produce data leakage, we provide a trace of a random message in the extended version of this paper (see [10]).

The data leakage is clearly visible in Figures 4(a), 4(c) and for the short message (see [10]). The attacks failed to recover a few of the leading bits de-



(a) Leading Constant Bit String message

(b) Bit sequence of LCBS Message

(c) Trailing Constant Bit String message

(d) Bit sequence of TCBS Message

Fig. 4. First 3ms of the exponentiation for LCBS and TCBS input messages. The system current was measured with a LeCroy 7-Zi digital oscilloscope [11] via low-side current shunt insertion. Figures 4(b) and 4(d) are a magnification of the time 2ms - 3ms for the respective input message.

pending on the class of input message, as described in Section 5. The system ran at 32MHz with no current limit and timing jitter enabled. Our experimental setup allowed us to capture the entire computation at this resolution. The system current was measured with a LeCroy 7-zi oscilloscope by performing a low-side shunt measurement over a 10Ω resistor.

Leading Constant Bit String: Figures 4(a) and 4(b) show the data leakage of the system while processing an LCBS input message. The LCBS message is the most important message class analyzed in this work, because any valid RSASSA-PKCS1-v1_5 message is a candidate LCBS message. On systems implementing highly optimized algorithms like ZDN, such as the smartcard system we analyzed, LCBS messages can also lead to data leakage despite the leading non-zero padding, i.e., leading `0xFF` (11111111_2) and `0xBB`(10111011_2). With ZDN, the look-ahead algorithm's sub-operations, `LABooth` and `LARed`, run in parallel and ensure that the system simply shifts across any leading 0- or 1-strings, deferring the bulk of the arithmetic operations, see Section 2.3. As a result, the constant structure of the leading bits of the message ensures a lower workload and lower

power dissipation during the multiply operation. These effects are clearly visible for the LCBS input message in Figure 4(a). We chose the message according to the scenario described in Section 3.2, i.e., we used the constant RSASSA-PKCS1-v1_5 padding and added a random 160-bit string as hash value. Messages coded in the ANSI x9.31 format resulted in very similar data leakage.

Trailing Constant Bit String: Figures 4(c) and 4(d) show the data leakage of the system while processing a TCBS input message. The TCBS message succeeds in inducing leakage despite the random padding used in the input message. The message consisted completely of random padding except for the least significant 160 bits. This illustrates the scenario described in Section 3.2, where the attacker is able to supply a hash value into the signature scheme directly. The 160 bits of SHA-1 make up only 16% of the entire message. However, the look-ahead algorithm's sub-operations, `LABooth` and `LARed`, ensure that the system simply shifts across any leading 0- or 1-strings and the trailing constant bit string of the input message is sufficient to induce leakage on our POC implementation. Note, our experiments show that a constant bit string consisting of as little as 5%-10% of the message is sufficient to induce data leakage. Thus, the attack would also work against 2048-bit RSA and for other common hash algorithms, such as SHA-256/384/512 or MD2 and MD5, respectively. Each of these hash functions accounts for at least 6.25% and 12.5% of the entire message, respectively, depending on whether 1024- or 2048-bit RSA is used.

Short Messages: As already mentioned in Section 3.2, short messages exploit both parts of parallel modular multiplication algorithms such as ZDN. In contrast to the square operation, during the multiply, because of the small value and low Hamming weight of the short message, the modular multiplication can be computed very quickly with very few iterations of the `LABooth` algorithm, see Section 2.3. In addition, since the intermediate result only grows by very few bits, reduction may potentially be eliminated entirely. If the partial product must be reduced, it can be computed with very few iterations of the loop during ZDN-based modular multiplication, see Figure 2.3. For these reasons the short message achieves the greatest difference in power consumption between squares and multiplies on the SLE78.

Potential Countermeasures: Because the attacks presented in this section require a particular structure, i.e., constant bit strings within the message, base blinding can defeat such attacks. However, it is worth noting that the "classical" blinding method as described by [9] actually fails in disrupting the constant bit string structure within the message. In this case, the message $m' := r^e \cdot m$ is used for the exponentiation instead of m, see Section 2.4. For common values of λ, i.e., 32-bit randoms, and small exponents, i.e., 3 or 17, the randomization introduced into the message is actually quite minimal. Additionally, the computation of the blinded message and its inverse becomes increasingly difficult for increasing λ's and exponents. For these reasons, an alternative form of base

blinding should be used, which ensures randomization of the entire message, namely $s = \left[(r_1 \cdot N + m)^d \bmod (r_2 \cdot N)\right] \bmod N$.

Exponent blinding could be used to obfuscate the exponent, however, blinded exponents can also be used to sign messages in the RSA signature scheme. However, its worth noting that RSA CRT exponents cannot be used to forge signatures, and for this reason we present several methods for recovering the full RSA private key from potentially blinded exponents in Section 5.

Techniques that decouple the execution from the data being processed, such as square-and-multiply-always, were able to prevent our attacks. However, other DPA countermeasures, such as timing jitter, had no affect in our analysis.

5 Full RSA Private Key Recovery

If an attacker can obtain a CRT exponent, which also might be blinded, he can not generate valid signatures with it since the CRT computation of signatures requires both p and q. Thus, the attacker must factorize the modulus N. We present three methods for the factorization of N of which only the first is known.

Lemma 1. *Let $N = pq$ be an RSA modulus, e the public exponent, and d the private exponent with $ed \equiv 1 \mod \varphi(N)$. Let $d_p = d \mod (p-1)$ and $d_q = d \mod (q-1)$ be the RSA private CRT exponents. Then given d_p or d_q, N can be factorized [2].*

Proof. Let $m < N$ be an arbitrary message. Without loss of generality, let d_p be known. Given (N, e) and d_p, we can compute $c = m^e \bmod N$ and $m_p = c^{d_p} \bmod N$. Since $p \mid N$ and $d_p = d \bmod (p-1)$, $m \equiv c^{d_p} \bmod p$ and $m_p \equiv c^{d_p} \bmod p$. Then $p = \gcd(N, m - m_p)$ [2].

Lemma 2. *Let $N = pq$ be an RSA modulus, e the public exponent, and d the private exponent with $ed \equiv 1 \mod \varphi(N)$. Let $d_p = d \mod (p-1)$ and $d_q = d \mod (q-1)$ be the RSA private CRT exponents. Then given a single blinded private CRT exponent $d_p{}' = d_p + r \cdot (p-1)$ or $d_q{}' = d_q + r \cdot (q-1)$, $r \in \mathbb{Z}$, N can be factorized.*

Proof. Let $m < N$ be an arbitrary message. Without loss of generality, let $d_p{}' = d_p + r \cdot (p-1)$, $r \in \mathbb{Z}$, be known. Given (N, e) and $d_p{}'$, we can compute $c = m^e \bmod N$ and $m_p = c^{d_p{}'} \bmod N$. Since $p \mid N$ and $d_p{}' = d + x \cdot (p-1)$ for some $x \in \mathbb{Z}$, $m \equiv c^{d_p{}'} \bmod p$ and $m_p \equiv c^{d_p{}'} \bmod p$. Then $p = \gcd(N, m - m_p)$.

We propose an elegant method for recovering the full RSA private key from blinded CRT exponents, which we call *exponent un-blinding*. This method spares the expensive modular exponentiations necessary in Lemma 2.

Lemma 3. *Let $N = pq$ be an RSA modulus, e the public exponent, and d the private exponent with $ed \equiv 1 \mod \varphi(N)$. Let $d_p = d \mod (p-1)$ and $d_q = d \mod (q-1)$ be the RSA CRT private exponents. Then given at least $k \geq 3$ blinded exponents $d_{p_i}{}' = d_p + r_i \cdot (p-1)$ or $d_{q_i}{}' = d_q + r_i \cdot (q-1)$, $r_i \in \mathbb{N}$, $i \in \{1, \ldots, k\}$, N can be factorized.*

Proof. Without loss of generality, let k blinded exponents d_{p_i}' be known. We calculate the pairwise differences $d_{p_{i,j}}' = |d_{p_i}' - d_{p_j}'| = |d_p + r_i \cdot \varphi(p) - (d_p + r_j \cdot \varphi(p))| = |r_i - r_j| \cdot \varphi(p) = r_{i,j} \cdot \varphi(p) = r_{i,j} \cdot (p-1)$, since $\varphi(p) = p-1$. Subsequently, we get $G = \gcd\{d_{p_{i,j}}'\} = \gcd\{r_{i,j}\} \cdot (p-1) = g \cdot (p-1)$. Thus, we can test whether $g = 1$, i.e., if $G = p-1$, by testing whether $N \equiv 0 \bmod (G+1)$. If fulfilled, we have found p and thus know the private key. Otherwise, we test whether $N \equiv 0 \bmod (G/g+1)$ for $g = 2, 3, 4 \ldots$.

Alternatively, if blinded exponents are easily obtainable, we simply obtain an additional blinded exponent $d_{p_{k+1}}'$ and perform the same calculations again. The higher k is, the higher is the probability of having a very small g. We provide experimental results and further information about the success probability in the extended version of this paper (see [10]).

The main advantage of the exponent un-blinding approach lies in the cheap computation costs. Many methods to obtain exponents, in fact, do not reveal the whole exponent [6, 15]. This also holds true for an attack using the short message presented in this work, since such attack cannot distinguish square and multiply operations during the computation of the first 7 - 8 bits, i.e., before the first reduction. This means that we cannot determine the value of these bits, apart from the most significant bit, which is always 1. Thus, an attacker has to test all the resulting possibilities. The complexity of these computations can be heavily reduced by applying exponent un-blinding. Although it requires at least three blinded exponents, the method is very efficient. The efficiency of the algorithm is two-fold: First, the required computations are cheap, especially compared to modular exponentiation, which is needed when the method presented in Lemma 2 is used. Second, a false guess for one of the exponents will quickly lead to an obvious false intermediate result, i.e., $G \ll N/2$. Thus, in contrast to the other methods, exponent un-blinding is able to handle more noise, more efficiently.

6 Conclusion

In this work, we presented two classes of messages that lead to data leakage. These classes are referred to as *leading constant bit string* (LCBS) and *trailing constant bit string* (TCBS). Valid input messages of common signature schemes, including PKCS#1 and ANSI x9.31, meet the criteria for these classes. Both classes and the *short message*, a specific LCBS message, were validated against an advanced smartcard system from Infineon, which ran our POC implementation of the RSA signature scheme. In all cases the input messages allowed for the recovery of the RSA private exponent in a single-trace SPA.

The short message exploits both multiplication and reduction of the modular multiplication in RSA signing. However, short messages can be prevented by means such as message length checks or padding. TCBS messages reiterate the importance of restricting direct user input. Our analysis showed that even if the most significant 95% of the message bits consist of random padding, if the attacker is able to set the least significant 5%, he will be able to recover

the secret data. Most importantly, LCBS messages demonstrate that even properly formatted messages can lead to distinct data leakage because of constant padding. For these reasons, we consider the constant paddings used by RSASSA-PKCS1-v1_5 and ANSI x9.31 to present a substantial security risk to modern cryptosystems that implement highly optimized algorithms, such as ZDN-based modular multiplication.

Our experimental results show that 0-strings result in far more distinctive data leakage than 1-strings. For this reason, zero padding should be avoided at all costs. In addition to non-constant padding, there are several countermeasures that can thwart such attacks, including square-and-multiply-always and a certain kind of base blinding. The initial reduction of RSA CRT also destroys any constant bit strings in the input message if the input message is larger than the modulus of the operation, i.e., larger than one of the prime factors of the modulus N. For these reasons, unless RSA CRT with initial reduction is used, we recommend that message blinding always be used on systems that implement a constant bit string padding scheme, such as the padding schemes of RSASSA-PKCS1-v1_5 and ANSI x9.31.

Additionally, a specific method for private key recovery when RSA CRT is used was presented, *exponent un-blinding*. This is substantially faster than the known methods if at least three distinct blinded exponents can be obtained. Most importantly, this method can cope with more noise, more efficiently and elegantly than any other known method.

In conclusion, this paper demonstrates that even an advanced cryptosystem, which implements recommended industry standards, can introduce additional unexpected side-channels. We believe that the claim "no attacks are known against RSASSA-PKCS1-v1_5" [18] is no longer true.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research, and by the Helmholtz Research School on Security Technologies.

The authors of this paper would like to thank all of their colleagues for their support. In particular we would like to thank Collin Mulliner and Christoph Bayer for their helpful and insightful input while writing the paper. We would also like to thank LeCroy Europe for their excellent technical support.

References

1. American National Standards Institute: ANSI X9.31-1998: Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry (rDSA) (1998)
2. Campagna, M., Sethi, A.: Key recovery method for CRT implementation of RSA (2004)
3. Courrège, J.C., Feix, B., Roussellet, M.: Simple Power Analysis on Exponentiation Revisited. In: CARDIS 2010. pp. 65–79 (2010)

4. Dhem, J.F., et al.: A Practical Implementation of the Timing Attack. In: Working Conference on Smart Card Research and Advanced Application. pp. 167–182 (1998)
5. Fischer, W., Seifert, J.P.: High-Speed Modular Multiplication. In: CT-RSA. pp. 264–277 (2004)
6. Halderman, J.A., et al.: Lest we remember: cold-boot attacks on encryption keys. Commun. ACM 52(5), 91–98 (2009)
7. Infineon Technologies AG: Contactless SLE 78 family: Next Generation Security. `http://goo.gl/qbQ30`
8. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Advances in Cryptology - Proceedings of Crypto '96. pp. 104–113. Springer-Verlag (1996)
9. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis: Leaking Secrets. In: Advances in Cryptology - Proceedings of Crypto '99. pp. 388–397. Springer-Verlag (1999)
10. Krämer, J., Nedospasov, D., Seifert, J.P.: Weaknesses in Current RSA Signature Schemes (Extended Version). `http://goo.gl/bu5MS` (2011)
11. LeCroy Corporation: WavePro 7 Zi Oscilloscope. `http://www.lecroy.com/Oscilloscope/OscilloscopeSeries.aspx?mseries=39`
12. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer-Verlag New York, Inc. (2007)
13. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press (1997)
14. Miyamoto, A., Homma, N., Aoki, T., Satoh, A.: Enhanced power analysis attack using chosen message against RSA hardware implementations. In: ISCAS. pp. 3282–3285 (2008)
15. Percival, C.: Cache missing for fun and profit. In: Proc. of BSDCan 2005 (2005)
16. Quisquater, J.J., Couvreur, C.: Fast decipherment algorithm for RSA public-key cryptosystem. Electronic Letters 18(21), 905–907 (1982)
17. Quisquater, J.J., Samyde, D.: Electromagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In: E-smart. pp. 200–210 (2001)
18. RSA: PKCS #1 v2.1: RSA Cryptography Standard. `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf` (2002)
19. Schindler, W.: A Timing Attack against RSA with the Chinese Remainder Theorem. In: CHES. pp. 109–124 (2000)
20. Sedlak, H.: Konzept und Entwurf eines Public-Key-Code Kryptographie-Prozessors (1985)
21. Sedlak, H.: The RSA Cryptography Processor. In: Advances in Cryptology — EUROCRYPT' 87, Lecture Notes in Computer Science, vol. 304, pp. 95–105. Springer Berlin / Heidelberg (1988)
22. Shamir, A.: Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attacks. US Patent 5991415 (23 Nov 1999)
23. Walter, C., Thompson, S.: Distinguishing Exponent Digits by Observing Modular Subtractions. In: Topics in Cryptology — CT-RSA 2001, LNCS, vol. 2020, pp. 192–207. Springer (2001)
24. Yen, S.M., Lien, W.C., Moon, S.J., Ha, J.: Power Analysis by Exploiting Chosen Message and Internal Collisions - Vulnerability of Checking Mechanism for RSA-Decryption. In: Mycrypt. pp. 183–195 (2005)