Template Attacks on ECDSA

Marcel Medwed[1] and Elisabeth Oswald[1,2]

[1] University of Bristol, Computer Science Department, Merchant Venturers Building, Woodland Road, BS8 1UB, Bristol, UK
[2] Graz University of Technology, Institute for Applied Information Processing and Communications, Inffeldgasse 16a, 8010 Graz, Austria
Elisabeth.Oswald@bristol.ac.uk

Abstract. Template attacks have been considered exclusively in the context of implementations of symmetric cryptographic algorithms on 8-bit devices. Within these scenarios, they have proven to be the most powerful attacks. This is not surprising because they assume the most powerful adversaries. In this article we investigate how template attacks can be applied to implementations of an asymmetric cryptographic algorithm on a 32-bit platform. The asymmetric cryptosystem under scrutiny is the elliptic curve digital signature algorithm (ECDSA). ECDSA is particularly suitable for 32-bit platforms. In this article we show that even SPA resistant implementations of ECDSA on a typical 32-bit platform succumb to template-based SPA attacks. The only way to secure such implementations against template-based SPA attacks is to make them resistant against DPA attacks.

1 Introduction

Template attacks are a fascinating research topic because they bring together statistical modelling and power analysis techniques. They consist of two phases. In the first phase, the attacker builds templates, i.e. statistical models for a device executing a certain sequence of instructions using fixed data. In the second phase, the attacker matches the templates with the traces acquired from the device under attack. The attacker chooses the templates for matching based on key hypotheses. The hypothesis that corresponds to the correct key always indicates the correct templates, and hence leads to the best matches. This allows determining the key.

The study of template attacks is also practically relevant because they essentially give a bound on the number of traces that are needed for a power analysis attack. This can serve as a measure for the resistance of a device against certain types of power analysis attacks.

Research on template attacks has concentrated mainly on their application to implementations of symmetric cryptographic algorithms. The reason for this is that the number of templates that need to be built, and the size of the templates, determines the practical feasibility of such attacks. Simply put, it is not obvious how to apply template attacks, such as they were described for implementations

of symmetric cryptographic algorithms on 8-bit platforms, to implementations of asymmetric cryptographic algorithms on more challenging platforms. We are not aware of any article tackling this problem to the best of our knowledge.

This article provides the first results on template attacks on implementations of asymmetric cryptographic algorithms on a 32-bit platform. To be more specific, we describe how a template-based SPA attack can be performed to break implementations of the elliptic curve digital signature algorithm (ECDSA). We describe why and how our attacks work, and how ECDSA actually helps to conduct these attacks. In addition to the theoretical description, we also show results of a practical implementation of our attacks on a 32-bit processor.

This article is organized as follows. In Section 2 we review previous work on template attacks. In Section 3 we briefly describe ECDSA and we explain why implementations of ECDSA are a particularly good target for template attacks. In Section 4 we describe the template-based SPA attack on our ECDSA implementation in detail. We summarize this contribution in Section 5.

2 Template Attacks

Template attacks exploit the fact that the power consumption of a device depends on the data it processes. The power consumption can be characterized by a multivariate normal distribution. In contrast to simple and differential power analysis (SPA and DPA), template attacks consist of two phases. In the first phase, the attacker builds the templates, i.e. characterizes the device. In the second phase, the templates are used for an attack. Template attacks can be used in an SPA and in a DPA scenario, see for instance [MOP07].

In this article, we focus on template attacks in an SPA scenario. We call these attacks template-based SPA attacks and make the following assumptions. During the characterization phase, the attacker has the opportunity to characterize the device under attack. This means, the attacker may invoke the instructions to be characterized with data of her choice. There is no limit on the number of invocations. In other words, the attacker has full control over the device. During the attack phase, we assume that the attacker has only very limited access to the device. Like in a single-shot SPA attack, we assume that the attacker only gets a single trace for an execution of ECDSA on the attacked device.

2.1 Template Building Phase

Characterizing the device (or the power consumption of a device) means determining the probability distribution of the power consumption of certain instructions. It is common to assume that the probability distribution of the power consumption is a multivariate normal distribution. This distribution is defined by a covariance matrix $\mathbf{C}$ and a mean vector $\mathbf{m}$, see (1).

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2 \cdot \pi)^n \cdot det(\mathbf{C})}} \cdot exp\left(-\frac{1}{2} \cdot (\mathbf{x} - \mathbf{m})' \cdot \mathbf{C}^{-1} \cdot (\mathbf{x} - \mathbf{m})\right) \qquad (1)$$

The covariance matrix $\mathbf{C}$ contains the covariances $c_{ij} = Cov(X_i, X_j)$ of the (interesting) points at time indices i and j. The mean vector $\mathbf{m}$ lists the mean values $m_i = E(X_i)$ for all (interesting) points in the trace. We refer to this pair $(\mathbf{m}, \mathbf{C})$ as *template* from now on.

As written before, we assume that we can determine templates for certain sequences of instructions. This means, we execute these sequences of instructions with different data d_i and keys k_j in order to record the resulting power consumption. Then, we group together the traces that correspond to a pair of (d_i, k_j), and estimate the mean vector and the covariance matrix of the multivariate normal distribution. As a result, we obtain a template for every pair of data and key (d_i, k_j): $\hbar_{d_i,k_j} = (\mathbf{m}, \mathbf{C})_{d_i,k_j}$.

In Mangard et al.[MOP07] the notation of reduced templates has been introduced. Reduced templates are templates for which the covariances can be neglected. In addition there are different strategies for template building described in [MOP07]. The different strategies lead to rather different complexities in the template building phase. In particular, building templates for intermediate values having already a power model in mind makes template attacks much more feasible in practice.

2.2 Template Matching Phase

During the template matching phase, we use the characterization together with a power trace from the device under attack to determine the key. This means, we evaluate the probability distribution function of the multivariate normal distribution with $(\mathbf{m}, \mathbf{C})_{d_i,k_j}$ and the power trace of the device under attack, i.e. we compute the probability:

$$p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_i,k_j}) = \frac{exp\left(-\frac{1}{2} \cdot (\mathbf{t} - \mathbf{m})' \cdot \mathbf{C}^{-1} \cdot (\mathbf{t} - \mathbf{m})\right)}{\sqrt{(2 \cdot \pi)^T \cdot det(\mathbf{C})}} \tag{2}$$

We compute this probability for every template. As a result, we get probabilities for all templates: $p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_1,k_1}), \ldots, p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_D,k_K})$. These probabilities measure how well the templates fit to a given trace. The highest probability indicates the correct template. Because each template is also associated with a key, we can derive the key that is used in the device.

2.3 Previous Work

The assumptions that we made before are the standard assumptions that are made in template attacks. Template attacks were introduced by Chari et al. [CRR03]. This original work has been investigated further by Rechberger and Oswald [RO04], who studied practical aspects of template attacks on implementations of RC4 on an 8-bit microcontroller. A result from these first two articles is that the choice and number of the so-called interesting points (i.e. the points in the power traces that are used to build the templates) is crucial for the success of template attacks. It turns out that an approach that works well in practice is to

choose points that lead to high correlation values in DPA attacks as interesting points.

In later work, Agrawal et al. [ARRS05] studied how to use template attacks to attack masking if the random number generator that produces the masks is biased during characterization. Archambeau et al. [APSQ06] investigated how principal subspaces can be used to make template attacks more effective. Oswald and Mangard demonstrated different types of template attacks on masked implementations of AES on an 8-bit smart card. They pointed out that in this particular scenario, template-based DPA attacks on a masked implementation are as effective as the same type of attack on an unmasked implementation. A comprehensive description of template attacks (in SPA and DPA scenarios) is given in Mangard et al. [MOP07].

2.4 Our Contribution

Previous work has solely focused on the application of template attacks to implementations of symmetric cryptographic algorithms such as AES and RC4. There seem to be two reasons for this.

The first and probably most important reason is that templates were often built for pairs of key and data. When considering asymmetric cryptographic algorithms this is obviously a problem because the numbers that are used in asymmetric cryptography are huge. Considering for instance elliptic curve cryptography, it is simply infeasible to build templates for all numbers in a finite field of size 2^{190} that can be used to represent coordinates of elliptic curve points. In addition, it is infeasible to build templates for all points on a given elliptic curve. In contrast, it is feasible to build templates for all 256 values that one byte of the AES state can take.

The second reason is that previous work was all experimentally verified on 8-bit microcontrollers. This is mainly because they are easy to get and to use for such experiments. To the best of our knowledge only the group of Gebotys have worked with 32-bit microprocessors. In order to study template attacks on asymmetric cryptographic algorithms and verify attacks experimentally, one needs to have a suitable measurement setup.

In this article we have found ways to overcome both problems. We are able to build templates for the intermediate points that occur in the ECDSA. Using these templates, we can break ECDSA implementations in different attack scenarios. We have verified our attacks experimentally on a 32-bit microprocessor. The microprocessor is based on an ARM7 architecture which is widely used in handheld computers, etc. Hence, our verification is done on a platform that is relevant in practice. This is the first work that successfully applies template attacks on ECDSA.

3 ECDSA and Power Analysis Attacks

In this section we explain why ECDSA is very well suited as target for template-based SPA attacks. First, we briefly describe ECDSA itself, and point towards

the combination of lattice attacks and simple power analysis. Then we review briefly common strategies to secure implementations of ECDSA. Last, we spell out three observations that can be combined in order to apply template-based SPA attacks to ECDSA implementations.

3.1 ECDSA

ECDSA is the elliptic curve version of the digital signature algorithm (DSA). This algorithm computes a signature, i.e. a pair of numbers (r, s), for a given message m. It is a non-deterministic algorithm because the computation of the r is based on an ephemeral key k. This ephemeral key changes in every signature operation. Hence, two signature operations on the same message m lead to two different signature values.

Essentially, the difference between ECDSA and DSA is in the calculation of the value r in the signature (r, s). This step involves operations on the specified elliptic curve using the ephemeral key k, see (3). The calculation of the value s makes use of the (static) secret key d, see (4). In (3) and (4), the variable P denotes the base point, and $h(m)$ denotes the hash value of the message m.

$$r = x_1 \pmod{n}, (x_1, y_1) = [k]P \tag{3}$$

$$s = k^{-1}(h(m) + d \times r) \pmod{n} \tag{4}$$

Breaking a signature scheme typically means either having the ability of producing valid signatures without knowing the secret key d, or, knowing the secret key d. The last meaning of breaking is often referred to as a full break.

For ECDSA, when the parameters are chosen appropriately, for instance by taking one of the recommended curves by NIST [Nat00], theoretical breaks are computationally infeasible. However, if for instance, the ephemeral key is known for a given signature (r, s) then the secret key d can be determined easily from (3) and (4). Even worse, it is sufficient if a small number of bits of the ephemeral key from a couple of signatures are known, lattice attacks can reveal the entire secret key d with low computational effort [NS03].

This is where SPA attacks come into play. If an attacker is able to reveal by SPA either the entire ephemeral key, or just a couple of bits of several ephemeral keys, ECDSA implementation can be broken in practice.

3.2 Security of Implementations of ECDSA

Implementations of ECDSA may allow revealing the secret key if they leak information about the ephemeral key or the secret key itself. Hence, the computation of r and s both need to be protected.

The computation of r makes use of the ephemeral key k. This ephemeral key is unknown to the attacker and it changes in each signature operation. Hence, DPA attacks are not possible on this operation, but only SPA attacks.

The computation of s makes use of the secret key d. If the attacker has knowledge about the messages being signed, DPA attacks are possible.

In this article, we focus on attacks on the computation of the r. It is well known that implementations that show visible differences between EC point doubling and EC point addition operations are highly susceptible to SPA attacks [Cor99]. There are abundant countermeasures against these types of SPA attacks. For instance, unified point operations or Montgomery-type multiplication techniques, see [Joy05] for a comprehensive overview. In addition, scalar blinding has been listed in [Cor99] to prevent SPA attacks.

3.3 Template-based SPA attacks on ECDSA

Countermeasures against SPA attacks might not prevent template-based SPA attacks because they exploit not only operation dependent leakages but also data dependent leakages. In this article we show that this is indeed possible for realistic implementations.

There are several observations or facts that when combined allow template-based SPA attacks on ECDSA implementations. The first observation, which we mentioned in the previous section, is that knowledge of only a small number of bits of several ephemeral keys of ECDSA signatures is sufficient to determine the secret key via lattice attacks. This is important because it means that not the entire ephemeral key needs to be determined.

The second observation, which is crucial for the practical realization is that the EC operation in the computation of r in ECDSA uses a fixed base point. This is important because it means that the first couple of bits of k that are processed during the computation of r can only lead to a small set of points (i.e. the multiples of P). This means that we only need to build templates for the points in this (small) set.

The third observation, which is also crucial for the practical implementation of template attacks, is that for software implementations of ECDSA on 32-bit processors, it is sufficient to build templates for intermediate values taking a suitable power model of the device into account. Especially for microprocessors previous work has shown that busses typically leak the Hamming weight (or sometimes the Hamming distance) of the operands. This leads to a dramatic simplification in practice. For instance, instead of building 2^{32} templates for a move instruction we only need to build 33 templates when we know that the device essentially leaks the Hamming weight.

These three observations make it clear that template-based SPA attacks are indeed feasible in practice: in order to determine the first f bits of the ephemeral key with a template-based SPA attack, we need to build templates to detect the 2^f multiples of the base point P. It is possible to build such templates because the set of points is rather small and because we can build templates that take the power model of the device into account.

In the next section we explain for a specific platform how two practical implementations of the EC point multiplication can be attacked by template-based SPA attacks using the three observations.

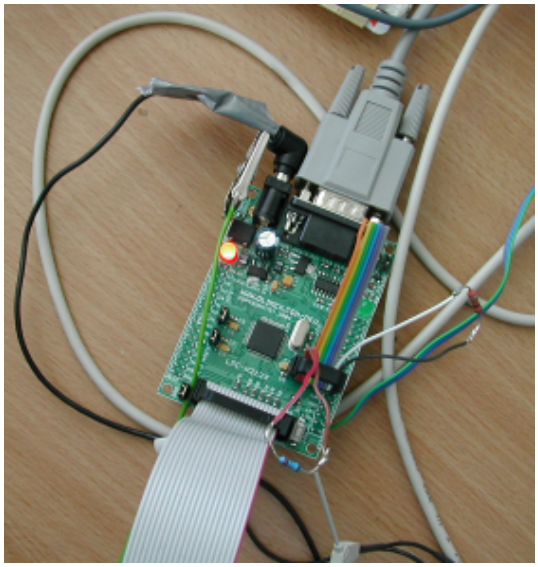

Fig. 1. Customized board with a 32-bit processor based on the ARM7 architecture

4 Practical Template-Based SPA Attacks on EC Point Multiplication

In order to demonstrate the feasibility of the attack strategy that we described in the previous section, we have implemented ECDSA on a suitable platform. One of the most widely used 32-bit processor architectures today is the ARM7. Hence, we have decided to use an ARM7-based platform for our practical experiments.

Our ECDSA implementation is optimized for the P192 NIST curve. We have taken care that the optimizations do not allow conducting SPA attacks using visual inspection: all finite field operations have data independent running time. We also make sure that variances that are due to microarchitectural features of the microprocessor, such as the pipeline or the early termination in the multiplier, do not lead to SPA vulnerabilities. To make sure that there are no SPA problems in the point multiplication algorithm we have implemented a typical double-and-always-add algorithm. We have also implemented window-versions of this algorithm. Summarizing, we have made sure that our implementation does not allow reading off the ephemeral key by a visual inspection. Hence there are no SPA leaks, but only DPA leaks in our implementation.

We can exploit these DPA leaks using template-based SPA attacks. In the subsequent sections, we first explain how we built the templates, and then we describe two scenarios to extract the ECDSA secret key by template-based SPA attacks on the ECDSA ephemeral key. Thereafter, we discuss the effectiveness of common countermeasures against our attacks.

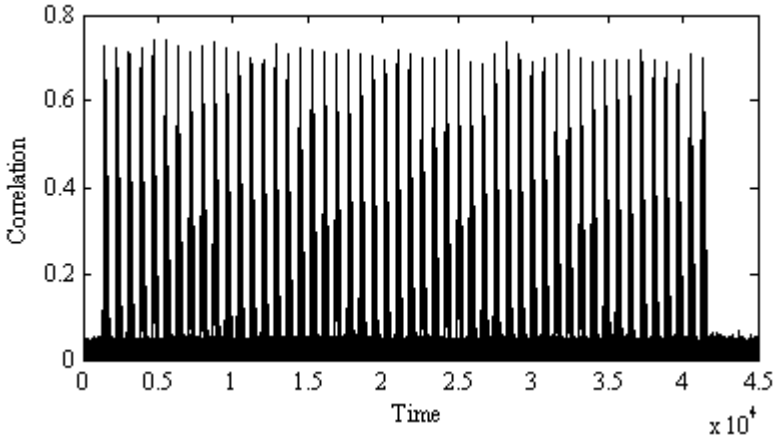

Fig. 2. Correlation coefficients of intermediate values of an EC operation

4.1 Template Building Phase

In order to find the interesting points for building templates, we performed DPA attacks on the EC point multiplication operation. This means, we executed the EC point multiplication algorithm several times with different input data and correlated the Hamming weight of a number of intermediate values to the power traces. Figure 2 shows the results of several such DPA attacks.

It turns out that in one step of a double-and-always-add algorithm, there are about 200 intermediate values that lead to high correlation coefficients. We have decided to select the points of interest by following the rule-of-thumb that Rechberger and Oswald gave in [RO04]: for each intermediate value that leads to a high correlation coefficient, we take the point in the power trace that leads to the highest absolute value in the correlation trace. This strategy helps to reduce the size of the templates because it restricts the number of point per intermediate value.

Nevertheless, having only this restriction still would lead to rather large templates. Hence we have decided to impose one further restriction on the selection of the points of interest. We have decided to select intermediate values that are "far away" (in terms of their Hamming distance) from their corresponding points in another template. Remember that we only have to build a small number of templates: if we build all templates before we start the actual attack (such as assumed in a classical template attack setting), we only need to build 2^f templates (one for each elliptic curve point that can occur at the beginning of the point multiplication algorithm). For each of these 2^f templates we have about 200 intermediate values that we can include. Some of these intermediate values will be very close in terms of their Hamming distance, while others will be farther away. The intermediate values that are farther away lead to templates

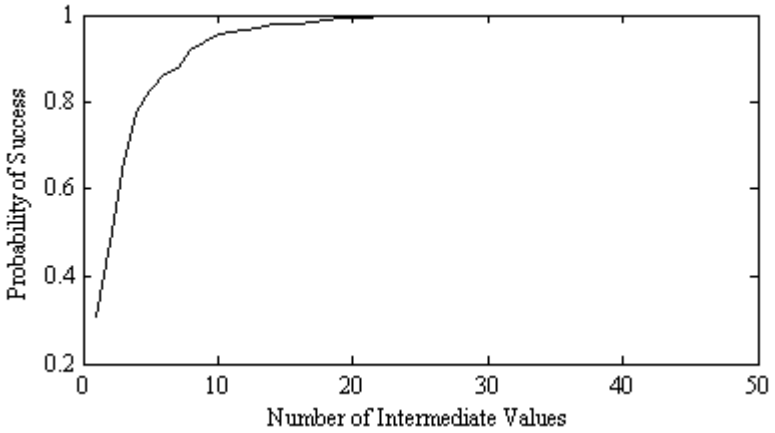

Fig. 3. Probability of success for an increasing number of intermediate values

that can be distinguished easier, because the mean vectors of their probability distributions are farther away.

With these considerations in mind, we have investigated the probability of success for the template matching for an increasing number of intermediate values. It turns out that for about 50 intermediate values (satisfying the property described before), the template matching succeeds with almost certainty, see Fig. 3.

4.2 Template Matching with Pre-computed Templates

In this section we describe a template-based SPA attacks with pre-computed templates. Pre-computing templates goes along with the assumptions that are traditionally made for template attacks.

In the scenario of attacking ECDSA implementations, the template matching phase includes acquiring one power trace for one ECDSA signature generation operation. The ECDSA operation is computationally expensive and hence acquiring a power trace (with reasonable quality) for an entire EC point multiplication might be not possible in practice. For instance, the memory of the data acquisition instrument might be too limited to store the entire trace. Hence, in practice we might only get a part of the power trace as input for a power analysis attack.

For our template-based SPA this limitation does not matter. We have pointed out already (for instance in Sect. 2.4), that lattice attacks require only a small number of bits of several ECDSA operations. Using template-based SPA attacks we can extract these bits step by step. In order to extract the first bit, we match the first two multiples of the base point with the corresponding part of the newly acquired power trace (if we attack a window method, we match the

first w points, where w corresponds to the window size). The template matching results in matching probabilities, see (2). Hence the template that leads to the highest probability indicates the correct bit of the ephemeral key. By applying this method iteratively, we can extract f bits of the ephemeral key. From then on we proceed with a lattice attack in order to derive the ECDSA secret key. Our practical attack using this strategy succeeds with probability close to one because our template matching succeeds with probability close to one.

4.3 Template Matching with On-the-fly created Templates

In all previous work, template attacks have been considered in the symmetric setting. As a consequence, the assumption has developed that template building always occurs before template matching. It appears that this is due to the fact that in the symmetric setting, it seems unlikely that the device under attack allows access to a known (and changeable key) and an unknown key at the same time. The asymmetric setting however is rather different. When we attack for instance an ECDSA signature generation operation with an unknown secret key, it is plausible to assume that we may invoke the ECDSA signature verification operation with (different) known public key(s).

This changes the situation for template building and matching. Now we can assume that we can interleave the template building with the matching steps. For instance, we can first use the ECDSA signature verification operation to build a template for a pair of EC points (again assuming a double-and-always-add algorithm). Then, we match the two points in order to extract one bit of the ephemeral key. Repeating this procedure for the remaining bits allows deriving the secret key using lattice attacks, such as in the attack that we described in the previous section.

4.4 Countermeasures

We have conducted our experiments on implementations of ECDSA that use SPA resistant point multiplication algorithms. Such algorithms do not prevent template-based SPA attacks because they only take care of SPA leaks (operation dependent leaks that are easy to identify using visual inspection).

However, also certain blinding techniques do not prevent our attacks. For instance, scalar blinding (i.e. using $k' = k + ran * ord(P)$, where ran is a random number) does not prevent our attacks. This is because k' and k are still congruent modulo the order of the base point and hence fulfill the ECDSA signing equation. This means, in a lattice attack only the size of the lattice grows, but the attack can still succeed if more bits of k' can be extracted via power analysis.

Point blinding (i.e. using $P' = ran * P$) does not necessarily protect against our attacks either. This is because the point blinding operation can be attacked using our technique in order to determine P'. With P' our original attack can be mounted in order to extract some bits of k. Hence, the attack is still feasible, although it requires a more effort in template building. This means, we either

build a larger set of templates before the attack, or we switch to our template creation on-the-fly technique.

As a matter of fact, the only types of countermeasures that prevent our template-based SPA attacks are countermeasures that provide DPA resistance by randomizing the coordinates of the base point P.

5 Conclusion

We have described template-based SPA attacks for implementations of ECDSA. It has turned out that such attacks can break implementations that are secure against other types of SPA attacks. We have combined three observations in our attacks. First, in order to determine an ECDSA secret key, only few bits of several ECDSA ephemeral keys need to be known. Second, ECDSA uses a fixed base point. Hence, we need to build templates only for a number of multiples of this base point. Third, for typical software implementations of ECDSA on microprocessors, it is sufficient to build templates for intermediate values taking the power model of the microprocessor into account. In our practical implementation of template-based SPA attacks, we have chosen templates according to criteria that increase the success probability in the matching phase such that this probability is close to one. Hence, our attacks succeed with almost certainty on our practical implementation. It turns out that our attacks can only be prevented by DPA countermeasures that randomize the coordinates of the base point. This article presents the first work on template attacks in the context of implementations of asymmetric cryptographic algorithms.

References

[APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[ARRS05] Dakshi Agrawal, Josyula R. Rao, Pankaj Rohatgi, and Kai Schramm. Templates as Master Keys. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2005.

[Cor99] Jean-Sébastien Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99, First International Workshop, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2003.

[Joy05] Marc Joye. *Advances In Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Note Series*, chapter V, Defences Against Side-Channel Analysis, pages 87–100. Cambridge University Press, 2005.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer, 2007. ISBN 978-0-387-30857-9.

[Nat00] National Institute of Standards and Technology (NIST). FIPS-186-2: Digital Signature Standard (DSS), January 2000. Available online at http://www.itl.nist.gov/fipspubs/.

[NS03] Phong Q. Nguyen and Igor E. Shparlinski. The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces. *Design, Codes and Cryptography*, 30(2):201–217, September 2003. ISSN 0925-1022.

[RO04] Christian Rechberger and Elisabeth Oswald. Practical Template Attacks. In Chae Hoon Lim and Moti Yung, editors, *Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers*, volume 3325 of *Lecture Notes in Computer Science*, pages 443–457. Springer, 2004.