# A Cautionary Note: Side-Channel Leakage Implications of Deterministic Signature Schemes

Hermann Seuschek[1]
hermann.seuschek@tum.de

Johann Heyszl[2]
johann.heyszl@aisec.fraunhofer.de

Fabrizio De Santis[1]
desantis@tum.de

[1]Technische Universität München, Institute for Security in Information Technology
Arcisstraße 21, 80333 München, Germany
[2]Fraunhofer Institute for Applied and Integrated Security
Parkring 4, 85748 Garching b. München, Germany

## ABSTRACT

Two recent proposals by Bernstein and Pornin emphasize the use of deterministic signatures in DSA and its elliptic curve-based variants. Deterministic signatures derive the required ephemeral key value in a deterministic manner from the message to be signed and the secret key instead of using random number generators. The goal is to prevent severe security issues, such as the straight-forward secret key recovery from low quality random numbers. Recent developments have raised skepticism whether e.g. embedded or pervasive devices are able to generate randomness of sufficient quality. The main concerns stem from individual implementations lacking sufficient entropy source and standardized methods for random number generation with suspected back doors. While we support the goal of deterministic signatures, we are concerned about the fact that this has a significant influence on side-channel security of implementations. Specifically, attackers will be able to mount differential side-channel attacks on the additional use of the secret key in a cryptographic hash function to derive the deterministic ephemeral key. Previously, only a simple integer arithmetic function to generate the second signature parameter had to be protected, which is rather straight-forward. Hash functions are significantly more difficult to protect. In this contribution, we systematically explain how deterministic signatures introduce this new side-channel vulnerability.

## Keywords

Elliptic Curve Cryptography; ECDSA; Deterministic Signatures; Side-Channel Attacks

## 1. INTRODUCTION

In contrast to classical signature schemes, deterministic signatures omit the requirement for high-quality and fresh randomness for each new signature generation to simplify secure implementations. In the area of low cost embedded systems such as small Internet of Things (IoT) devices, high-quality true random number generators are not generally available. Using deterministic signatures simplifies the implementation of digital signatures on such devices because high-quality entropy sources are no longer needed. Another practical advantage is that the deterministic behavior increases testability and verifiability of implementations which enhances the quality and as a result the security of implementations.

In this work, we focus on the ECDSA standard [3] as an example, however, all properties and conclusions also apply to similar schemes like ECGDSA [18], ECKDSA [18], and general EC-El-Gamal [16] signatures. The generation of ECDSA signatures consist of two computations leading to the two signature components $R$ and $S$. In the following, $k$ is the randomly chosen ephemeral secret key, $P$ is the constant elliptic curve base point, with $n$ the base point order, $(x_R, y_R)$ are the coordinates of the resulting point after the Elliptic Curve Scalar Multiplication (ECSM) $k \cdot P$, the hash value of the message $m$ is denoted by $\mathrm{H}(m)$, and $d$ is the long-term secret key. The computation of a signature works as follows:

$$\text{randomly select } k \in \{1, \dots n-1\} \tag{1}$$

$$(x_R, y_R) = k \cdot P \tag{2}$$

$$R = x_R \bmod n \tag{3}$$

$$S = k^{-1} \cdot (\mathrm{H}(m) + d \cdot R) \bmod n \tag{4}$$

Attackers usually either aim to recover the long-term secret key $d$ or want to forge signatures for arbitrary messages. To recover the long-term secret key $d$, an adversary can either target the secret key $d$ directly or the ephemeral secret $k$. Both are of equal value because $d$ can easily be computed from $k$ for a given signature as

$$d = R^{-1} \cdot (S \cdot k - \mathrm{H}(m)) \bmod n. \tag{5}$$

Using the same ephemeral key for at least two signatures also causes a major threat. For two signatures $(R, S)$ and $(R, S')$ of different messages $m$, $m'$, the private key can be easily extracted. First, the difference of the two signature values $S$ and $S'$ is computed:

$$S - S' = k^{-1} \cdot (\mathrm{H}(m) - \mathrm{H}(m')) \bmod n \tag{6}$$

This difference allows the computation of the ephemeral key

$$k = (S - S')^{-1} \cdot (\mathrm{H}(m) - \mathrm{H}(m')) \bmod n, \tag{7}$$

*This is the accepted version of a publication at CS2 '16, January 20 2016, Prague, Czech Republic. Final publication can be found under under DOI:* http://dx.doi.org/10.1145/2858930.2858932

which allows the computation of the long term secret $d$ by using Equation 5. This method was used for breaking the code signing mechanism of Sony Playstation 3 [13], where a *constant value* has been used as the ephemeral key leading to a recovery of the secret signature key. Nguyen et al. [25] proposed an even more sophisticated attack based on lattices which only requires the knowledge of few bits of $k$ to completely recover $k$ (and as a consequence $d$).

Importantly, these facts underline that the security of the long term secret key $d$ directly depends on the quality, respectively entropy, of the chosen random number for the ephemeral key $k$. This was the starting point for the idea of deterministic signature schemes. In order to eliminate the random number, the ephemeral key is derived from the private long term signature key which necessarily has sufficient entropy, and the message to be signed. Using the private key in conjunction with the message makes sure that the ephemeral key cannot be derived from the message and that it is unique for each message. A hash function is already required for signature generation. Hence, it is reasonable to reuse it to derive the ephemeral key in a message authentication code (MAC) construction such as HMAC [1]. The deterministic derivation of the ephemeral key has no influence on the verification of such signatures. Hence, this modification of the generation side can easily be used in existing installations and verifiers have no means of detecting whether a deterministic method had been used.

While we think that deterministic signatures have a significant value, especially in IoT devices, we are concerned about the *side-channel implications*, especially in the context of the same IoT devices. The additional usage of the long-term secret key $d$ in a hash function with a second input being the plain message, which is varying and known to possible attackers, enables powerful and hard to prevent Differential Power Analysis (DPA) attacks. In this contribution, we would like to explain this *introduced side-channel vulnerability* on the example of two recently proposed schemes by Bernstein et al. [6] and Pornin [26].

Our paper is structured as follows: in Section 2 we discuss side-channel vulnerabilities of classical ECDSA signatures. In the subsequent Section 3 we will give an introduction to deterministic signatures and discuss the vulnerabilities of the deterministic derivation of the ephemeral key. Finally, in Section 4 we conclude findings and give suggestions for further directions of work.

# 2. SIDE-CHANNEL ATTACKS ON ECDSA IMPLEMENTATIONS

In this section we discuss side-channel vulnerabilities for conventional elliptic curve-based signature schemes, where the ephemeral keys are selected at random. Attacks generally either target the ephemeral scalar $k$, or the secret key $d$. (The random number generators are generally not considered.) Since the ECDSA scheme uses an individual ephemeral scalar $k$ for each signature, the Elliptic Curve Scalar Multiplication (ECSM) must only be protected against single-execution attacks, which generally is a big advantage. However, several kinds of single-execution attacks like SPA or profiled template attacks have to be considered. To protect against SPA [20], a constant processing time and constant sequence of operations, independent of the processed scalar provides sufficient protection. This can be achieved through applying the Montgomery powering ladder [19, 22], the Double-and-add-always algorithm [11], and through unified formulas for doubling and adding [8]. To protect against template attacks [24], the required profiling for these attacks can be defeated through base point blinding [11], scalar randomization [11], random scalar splitting [14], random field representations [9], or projective coordinate randomization [11]. Those mechanisms provide a good level for protection and are straightforward to implement. A drawback is that randomness is required. However, the required entropy for this side-channel protection is low compared to what is required for ephemeral keys because no offline computation must be considered. Even if profiling can be prevented as described, more sophisticated *non-profiled* single-execution attacks remain. They include the Big-Mac attack by Walter [28] and recent improvements [10, 15, 12]. Their impact is, in principal, only limited by the available single-execution leakage and its signal to noise ratio. Still, such attacks may only employ single executions at once which limits their threat. As a summary, the ECSM operation can usually be protected against side-channel attacks up to a fair security level with reasonable effort.

Alternatively, attackers may target the regular long number multiplication $d \cdot R$ in Equation 4 to recover the secret key $d$. An attack targeting this operation on a hardware implementation of ECDSA was reported in [17]. A possible countermeasure against this attack is the transformation of Equation 4 in a way such that the publicly known value $R$ is not directly multiplied with the private key $d$ to prevent differential attacks with known inputs:

$$s = k^{-1} \cdot \mathrm{H}(m) + r \cdot (k^{-1} \cdot d) \bmod n. \tag{8}$$

Additionally, the multiplications can be protected using masking.

# 3. SIDE-CHANNEL ATTACKS ON DETERMINISTIC SIGNATURE SCHEMES

The basic idea of deterministic signature schemes is to compute the ephemeral key by processing the message to be signed combined with the private key. While the private key is constant, the message is known to attackers and changes for each computation of a new ephemeral key. Both, the secret key, and the known message are inputs to a hash function. Unfortunately, these are the exact circumstances which enable powerful *differential side-channel attacks*. Compared to the conventional case, differential attacks are now possible on a hash function implementation which, contrary to the regular multiplication discussed above, is much harder to protect against such attacks. Before we discuss the side-channel vulnerabilities of deterministic signature schemes proposed by Bernstein and Pornin more in detail, we start with an introduction to attacks on the hash functions SHA-2 and SHA-3, which form the basic components for the deterministic ephemeral key generation.

## 3.1 Side-Channel Attacks on SHA-2

McEvoy et al. [23] and more recently Belaïd et al. [5] report successful side-channel attacks on a HMAC construction based on the SHA-2. The HMAC construction invokes the underlying hash function $\mathrm{H}(\cdot)$ in a nested way, with the consequence that the side-channel attack has to be performed two times, for the inner and outer hash. The

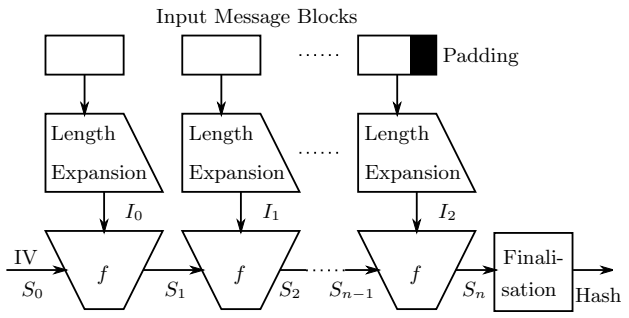


**Figure 1: Merkle-Damgård construction is a method to build a cryptographic hash function from an one-way compression function $f$.**

HMAC-values for a message $M$ using a key $K$ is computed as follows, where *opad* and *ipad* are standardized padding values with the same length as the input block size, and | denotes concatenation:

$$\mathrm{HMAC}(K, M) = \mathrm{H}\left((K \oplus opad)|\mathrm{H}((K \oplus ipad)|M)\right) \quad (9)$$

One of the proposed deterministic signature schemes utilizes a deterministic random bit generator based on HMAC where the HMAC-key $K$ is initially known and updated by an invocation of the HMAC. In this case, the required side-channel attack can be reduced to an attack of the underlying hash. Therefore we are discussing the attack methodology for the hash function only. The attack is based on properties of the Merkle-Damgård construction and therefore applicable to all hash functions which are designed according to Merkle-Damgård (e.g. MD-5, SHA-1, SHA-2, ...). This construction technique for hash functions is depicted in Figure 1. The input message is split into equally sized input blocks. In order to allow the hashing of arbitrary length messages, a proper padding of the message has to be applied. The resulting input blocks are expanded by a length expansion and compressed by the function $f$ in a chained way. Therefore, the compression function has one output and two inputs, one with the size of the expanded message block and one with the same size as the output. The compression function for the first message block is initialized by an initialization vector $IV$, which is the initial internal state $S_0 = IV$. The output $S_n$ of the final execution of the compression function can be optionally post-processed to form the final hash value.

The side-channel attack described in [23] and improved in [5] targets the compression function with the prerequisite that the input message to the hash function has to be composed of a fixed and eventually padded key value which fits into the first input block $I_0$ of the hash, and a varying part such as an arbitrary message. As a consequence the output $S_1$ of the first execution of the compression function $f$, which is the input of the second one, is fixed and unknown. Taking a closer look on the second execution of the compression function we have a fixed unknown input $S_1$ and a known varying input block $I_1$. This enables the possibility to perform differential side-channel attacks which can derive the internal state $S_1$. It is important to note that this attack does not reveal the first unknown input block (e.g. the secret key) but an internal value which allows the computation of the correct hash value of arbitrary messages with respect to the secret key. For the HMAC setting according to Equation 9, this value is used as a known input for the outer hash function which has to be attacked in the same way. The same attack methodology will be applied to the actual deterministic signature schemes in Sections 3.3 and 3.4.

## 3.2 Side-Channel Attacks on SHA-3

Taha and Schaumont [27] report different levels of side-channel vulnerabilities of MAC-Keccak depending on the selected Keccak parameters and the key length.

We discuss the subset standardized as SHA-3 [2] and give a brief introduction to it. For a more detailed description we refer to [7, 2]. The hash operation is based on a so-called sponge functions and the analogous terminology of absorbing and squeezing is used. It is performed in three high-level steps:

1. Like in the SHA-2, the input message is *padded* and split into equally sized blocks. The size of the blocks is called the rate $r$ and equals the bit-length of the SHA-3 digest. The initial state is set to all zero bits.
2. The 3-dimensional state of size 1600 bits, see Figure 2, is filled from the bottom to the top starting from position $(x = 0, y = 0, z = 0)$ in z-direction. When a full input block with size r bits is *absorbed* by the state, the remaining bits are filled with zeros. The number of zero bits is referred as capacity $c$. The resulting state is XORed with the previous state and applied to the Keccak function to update the state.
3. The final hash digest equals the first $o$ bits in the state which are *squeezed* out of the state.

In case of SHA-3, the Keccak function consists of 24 rounds of five sequential operations. Where each round is defined as

$$Output = \iota \circ \chi \circ \pi \circ \rho \circ \theta(Input). \quad (10)$$

For side-channel attacks on SHA-3 we are only interested in the knowledge of $\theta$, therefore we only present this operation which is defined as

$$\begin{aligned} d[x][y][z] = d[x][y][z] \oplus (\oplus_{i=0}^{4} s[x-1][i][z]) \\ \oplus (\oplus_{i=0}^{4} s[x+1][i][z-1]). \end{aligned} \quad (11)$$

The $\theta$ operation given in Equation 11 is the binary XOR of each bit in the state with itself and the bits of two neighbor columns as depicted in Figure 2. The attack target is the XOR-operation between the bits in each state-column. For successful attacks, the constant secret bits and the known varying bits have to be stored in a certain configuration in the state. That is at maximum one secret bit per column and for each secret bit at least one varying bit in the same column.

## 3.3 Edwards-curve Digital Signature Scheme

The digital signature scheme proposed by Bernstein et al. [6] is a variant of ECDSA signatures. It is gaining popularity in embedded applications due to fact that no license fees are due and that open source implementations are available which provide all functions required to establish sound information-secure devices. The scheme is based on twisted

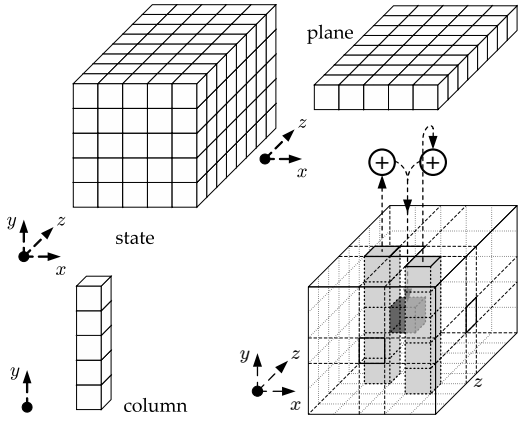


**Figure 2: SHA-3 state and the $\theta$ operation (taken from keccak.noekeon.org and modified in accordance to CC BY 3.0)**

Edwards curves and called Edwards-curve Digital Signature Scheme (EdDSA). The long term secret key $d$ is of the length of the curve parameter $b$ which is 256 bits. The selection of the hash function is highly dependent on this parameter $b$ because of the required output size of $2b$ bits. The recommended hash function by Bernstein [6] is SHA-512 with the option to be later replaced by SHA3-512. We restrict our analysis to vulnerabilities caused by using these particular hash functions which are explained in the following subsections.

The long term private signature key $d$ is a randomly chosen b bit integer. In order to compute the corresponding public key, the private key has to be hashed to obtain the 2b bit value $\mathrm{H}(d) = (h_0, h_1, \ldots, h_{2b-1})$. One half of this hash value is used to derive $a = 2^{b-2} + \sum_{3 \leq i \leq b-3} 2^i h_i$, which is used to compute the public key by multiplying it with the base point of the elliptic curve $A = a \cdot P$. The remaining part of the hash value is used to derive the deterministic ephemeral key $k$ by concatenating it with the message to sign $m$ (denoted by $|$) and hashing the resulting value as follows:

$$k = \mathrm{H}(h_b, h_{b+1}, \ldots, h_{2b-1}|m). \tag{12}$$

The signature of the message $m$ is the pair $(R, S)$ where $R = k \cdot P$ and $S = (k + \mathrm{H}(R|A|m)a) \bmod l$; the modulus $l$ is a prime number between $2^{b-4}$ and $2^{b-3}$.

The derivation of the ephemeral key according to Equation 12 bears side-channel risks due to the fact that a successful side-channel attack allows the computation of the corresponding $k$ without knowing the private key $d$. As a result valid signatures can be computed. In the following we will discuss the side-channel vulnerability for the two recommended hash functions.

Side-Channel Attack on the SHA-512 Variant The most important parameters to perform a side-channel attack on Ed25519 using SHA-512 is the size of the input blocks which is 1024 bit and the size of the unknown key dependent value $(h_b, h_{b+1}, \ldots, h_{2b-1})$ which is 256 bit. This means that only the first input block of the hash depends on an unknown value and the attack described in Section 3.1 is possible. The side-channel attack allows to determine the input value of the second invocation of the compression function $S_1$ which is the same as the output of the first block compression. With this result it is possible to compute a corresponding ephemeral key for a message with the *restriction* that the first 768 bit of the message have to be constant. This is because this part of the message is processed by the first invocation of the compression function together with the unknown key-dependent value. In practice this can be exploited when files with constant header information are signed.

Side-Channel Attack on the SHA3-512 Variant As the MAC-Keccak construction is exactly the same as the derivation of the ephemeral key in Ed25519 (i.e. an unknown value concatenated with a known manipulatable message is hashed), we can directly apply the results from Taha and Schaumont [27] to our analysis of the signature scheme. Unlike in the work of Taha and Schaumont, in our case the parameters are fixed to the following values:

- The state size is fixed to 1600 for all SHA3 variants
- The plane $p$ is 360 bits long for SHA3-512
- The output length $o$ is 512 bit
- The rate $r$ (size of one input block) is 576 bit
- The length $b$ of the unknown value is 256 bit

According to [27] the side-channel attack can be performed in the most simplest way if $b \leq p$, which is given in our setting. In this case the internal state of SHA-3 will absorb the key only in the bottom plane (see Figure 2) the remaining bits of the state are filled with known message bits and zeros. The resulting state is XORed with the initial state which is all zeros and thus has no effect on the state. Afterwards the Keccak function is applied to the state beginning with the $\theta$-operation. While computing this operation every column of the state consists of at most one unknown key bit. So the first step of the computation on $\theta$ is sufficient to recover the unknown value. The resulting value is not the private key itself, but knowing this internal value allows the computation of ephemeral keys for arbitrary messages.

To understand the attack on $\theta$ we give a brief description of the computations performed in typical implementations. According to [27], the $\theta$-operation is computed in two steps. In the first step the XOR of all columns in the state is computed resulting in $\theta_{plane}$. The XOR for each column can be computed in parallel in hardware, or from bottom to top like in the reference software implementation. Alternative implementations can also use a top to bottom approach. In our case all three implementations variants can be attacked by a side-channel attack because $b \leq p$. In the second step, every bit in the state is XORed with the two neighbor locations of $\theta_{plane}$. By executing several signatures with the unknown private key and changing known messages a differential side-channel attack can be mounted to reveal the secret bits $(h_b, h_{b+1}, \ldots, h_{2b-1})$.

## 3.4 Signatures According to RFC 6979

In his Request for Comments [26], Pornin describes an alternative deterministic derivation of the ephemeral $k$ for ECDSA. The described method is based on HMAC-DRBG (Deterministic Random Bit Generator) as specified by NIST [4]. The underlying HMAC is defined in [21, 1] as shown in Equation 9. Here, the utilized cryptographic hash function is denoted by $\mathrm{H}(\cdot)$ which is the same as used for the signature itself.

In the HMAC-DRBG construction, the underlying HMAC is executed for several times. The initial input values are $K_0 = 0$ for the HMAC-key and $V_0 = (0x01, 0x01, \ldots, 0x01)$ with length $\lceil \frac{hlen}{8} \rceil$ bytes, where the length of the hash output is denoted by *hlen*. The initial single byte $F_0$ is set to zero. The first invocation of the HMAC updates the key $K_i$ in the following way, where $i = 1$:

$$K_i = \text{HMAC}(K_{i-1}, (V_{i-1}|F_{i-1}|d|\text{H}(m))) \tag{13}$$

The input message is a concatenation of the initial $V_0$, the initial $F_0$, the ECDSA private key $d$ and $\text{H}(m)$ which is the hash value of the message to sign. To understand the details of the initial execution of HMAC according to Equation 13 it can be rewritten as follows, substituting Equation 9 into Equation 13:

$$K_i = \text{H}\left((K_{i-1} \oplus opad)|\text{H}((K_{i-1} \oplus ipad)|V_{i-1}|F_{i-1}|d|\text{H}(m))\right) \tag{14}$$

The input value of the *inner hash* is a concatenation of several known variable and constant values with the constant and unknown private key $d$. This is the basic requirement to perform differential side-channel attacks. After updating the key value $K_i$, the value $V_i$ is updated which can be computed with known values:

$$V_i = \text{HMAC}(K_i, V_{i-1}) \tag{15}$$

In the next iteration, the values $K_i$ and $V_i$ are updated again to finally get $K_2$ and $V_2$. The update to $K_2$ is the same as given in Equation 13 with the difference that byte $F_1 = 1$. The update of value $V_i$ is the same as in Equation 15. The computation of the ephemeral key is done based on the resulting $K_2$ and $V_2$ with the message $m$ and private key $d$ not involved in the computation. This means, that retrieving $K_2$ and $V_2$ from a side-channel attack allows the computation of the ephemeral key $k$. In this case the private key $d$ is disclosed and an attacker is able to sign arbitrary messages. In the following we will discuss the side-channel vulnerability dependent on the selection of the underlying hash function and the elliptic curve parameters. Thereby we will concentrate on *attacking the inner hash* of the HMAC as the outer hash can simply be computed because of the knowledge of $K_i$. Nevertheless the side-channel attack has to be performed twice because of the differing values of $K_i$, $V_i$, and $F_i$, where $i \in \{0, 1\}$:

$$\text{H}((K_i \oplus ipad)|V_i|F_i|d|\text{H}(m)) \tag{16}$$

The value $K_i \oplus ipad$ is known and has exactly the size of one input block of the hash function. Hence, the internal state after processing the first block can be derived from known values. This means that the first block does not have to be considered for side-channel analysis.

Next we have to consider which values are processed for the second block depending on different combinations of hash functions and curve parameters.

Side-Channel Attack on the SHA-2 Variant Analyzing the use of SHA-2 variants while using the attack methods proposed in Section 3.1 we have to make sure that the input data composed of $V_i$, $F_i$, and $d$ exactly fits into the second input block. Then the side-channel attack according to [5] can be performed on the input of the third compression function which is the same as the output of the second compression function $S_2$. When the three values $V_i$, $F_i$, and $d$ get larger than the second block, the private key $d$ spreads to the third block which prevents the side-channel attack. When it is to small, the input value of the second compression cannot be constant because the output bits of the hash $H(m)$ cannot be controlled.

**Table 1: Comparison of the input block length *blen* and the size of the concatenation of $V_i$, $F_i$, and $d$.**

| | *blen* | P-192 | P-224 | P-256 | P-384 | P-521 |
|---|---|---|---|---|---|---|
| SHA-224 | 512 | 424 | 456 | 488 | 616 | 753 |
| SHA-256 | 512 | 456 | 488 | 520 | 648 | 785 |
| SHA-384 | 1024 | 584 | 616 | 648 | 776 | 913 |
| SHA-512 | 1024 | 712 | 744 | 776 | 904 | 1041 |

Table 1 compares the block sizes and the length of the concatenation of $V_i$, $F_i$, and $d$ for different hash and ECC NIST curve combinations, regardless of whether the combinations are recommendable. As it can be seen, the requirement for matching block length *blen* and input sizes is not met for all combinations which essentially prevents side-channel attacks on this construction.

Side-Channel Attack on the SHA-3 Variant For SHA-3 the analysis is more complex because of the characteristics of the sponge construction. Therefore we discuss the possible vulnerabilities on an exemplary hash/curve combination with a P-256 curve and the SHA2-256 hash function.

We have to analyze whether the values $V_i$, $F_i$, $d$, and $H(m)$ are stored in the SHA-3 state, such that the side-channel attack sketched in Section 3.2 is applicable. For the given sizes of the values, all bits fit into one hash input block. The private key is shorter than the size of one plane (256 bit vs. 320 bit) and therefore only one secret bit is stored per column. The hash value of the message $H(m)$ is of the same length as the secret. This means that for attacking all bits of the secret, both values have to be aligned in a way that each column with a secret key bit also stores a hash values of the message. This is not possible with the given setting which prevents a side-channel attack targeting all secret bits. Nevertheless, there is an overlap which allows the recovery of some bits of the key $d$.

# 4. CONCLUSION

In this paper we discussed side-channel vulnerabilities arising from the introduction of deterministic signatures. The key novelty for this kind of signature generation is the fact that the secret key and the message are processed in a hash function. This enables differential side-channel attacks. Unfortunately, hash functions are difficult to protect against differential side-channel attacks. Hence, the advantages of deterministic ephemeral keys not requiring a true random number generator come with significant drawbacks for implementations in IoT and similar applications, where side-channel attacks are possible.

The precise extent of the vulnerability depends on the selected signature scheme, elliptic curve, and hash function. The scheme proposed by Bernstein seems to be more vulnerable to side-channel attacks caused by using the full message to derive the ephemeral key. In contrast, the HMAC-DRBG construction used by Pornin's proposal offers more security by higher attack complexity and only using a hash value of the message.

A possible direction for future work is to perform exper-

iments which support our results and help to quantify the vulnerability for different configurations. Additionally, appropriate countermeasures have to be developed.

## Acknowledgments

This work was partly funded by the German Federal Ministry of Education and Research in the project SIBASE through grant number 01IS13020.

## 5. REFERENCES

[1] FIPS PUB 198-1: The Keyed-Hash Message Authentication Code (HMAC). Technical report, Information Technology Laboratory, National Institute of Standards and Technology, July 2008.

[2] FIPS PUB 202: SHA-3 Standard, Permutation-Based Hash and Extendable-Output Functions. Technical report, Information Technology Laboratory National Institute of Standards and Technology, Aug. 2015.

[3] ANSI. *ANS X9.62-2005. Public Key Cryptography for the Financial Services Industry. The Elliptic Curve Digital Signature Algorithm (ECDSA)*. American National Standards Institute, 2005.

[4] E. Barker, J. Kelsey, et al. Nist special publication 800-90a: Recommendation for random number generation using deterministic random bit generators, 2012.

[5] S. Belaid, L. Bettale, E. Dottax, L. Genelle, and F. Rondepierre. Differential power analysis of HMAC SHA-2 in the Hamming weight model. In *Security and Cryptography (SECRYPT), 2013 International Conference on*, pages 1–12. IEEE, July 2013.

[6] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, 2012.

[7] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. The Keccak SHA-3 submission, January 2011. http://keccak.noekeon.org/.

[8] E. Brier and M. Joye. Weierstraß Elliptic Curves and Side-Channel Attacks. In D. Naccache and P. Paillier, editors, *Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 335–345. Springer Berlin Heidelberg, 2002.

[9] M. Ciet and M. Joye. (virtually) free randomization techniques for elliptic curve cryptography. In *Information and Communications Security*, volume 2836 of *Lecture Notes in Computer Science*, pages 348–359. Springer Berlin / Heidelberg, 2003.

[10] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil. Horizontal Correlation Analysis on Exponentiation. In M. Soriano, S. Qing, and J. López, editors, *Information and Communications Security*, volume 6476 of *Lecture Notes in Computer Science*, pages 46–61. Springer Berlin Heidelberg, 2010.

[11] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *CHES '99: Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems*, pages 292–302, London, UK, 1999. Springer-Verlag.

[12] J.-L. Danger, S. Guilley, P. Hoogvorst, C. Murdica, and D. Naccache. Improving the Big Mac Attack on Elliptic Curve Cryptography. Cryptology ePrint Archive, Report 2015/819, Aug. 2015.

[13] Fail0verflow. Console Hacking 2010 – PS3 Epic Fail. https://events.ccc.de/congress/2010/Fahrplan/events/4087.en.html, Dec. 2010.

[14] P.-A. Fouque and F. Valette. The doubling attack - why upwards is better than downwards. In C. Walter, Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 269–280. Springer Berlin / Heidelberg, 2003.

[15] J. Heyszl, A. Ibing, S. Mangard, F. De Santis, and G. Sigl. Clustering Algorithms for Non-profiled Single-Execution Attacks on Exponentiations. In A. Francillon and P. Rohatgi, editors, *Smart Card Research and Advanced Applications*, volume 8419 of *Lecture Notes in Computer Science*, pages 79–93. Springer International Publishing, 2014.

[16] P. Horster, H. Petersen, and M. Michels. Meta-ElGamal signature schemes. In *CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security*, pages 96–107, New York, NY, USA, 1994. ACM.

[17] M. Hutter, M. Medwed, D. Hein, and J. Wolkerstorfer. Attacking ECDSA-Enabled RFID Devices. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *Applied Cryptography and Network Security*, volume 5536 of *Lecture Notes in Computer Science*, pages 519–534. Springer Berlin Heidelberg, 2009.

[18] ISO. *ISO/IEC 15946-2: Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: Digital Signatures*. International Organization for Standardization, 2002.

[19] M. Joye and S.-M. Yen. The montgomery powering ladder. In B. Kaliski, Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin / Heidelberg, 2003.

[20] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, 1999. Springer-Verlag.

[21] H. Krawczyk, R. Canetti, and M. Bellare. RFC2104 - HMAC:Keyed-Hashing for Message Authentication, 1997.

[22] J. López and R. Dahab. Fast multiplication on elliptic curves over GF(2m) without precomputation. In *CHES '99: Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems*, pages 316–327, London, UK, 1999. Springer-Verlag.

[23] R. McEvoy, M. Tunstall, C. C. Murphy, and W. P. Marnane. Differential power analysis of hmac based on sha-2, and countermeasures. In *Information security applications*, pages 317–332. Springer, 2007.

[24] M. Medwed and M. E. Oswald. Template attacks on ECDSA. In *9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers*, Lecture Notes in Computer Science, pages 14 – 27. Springer, 2009.

[25] P. Q. Nguyen and I. E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Designs, codes and cryptography*, 30(2):201–217, 2003.

[26] T. Pornin. Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA). RFC 6979 (Informational), Aug. 2013.

[27] M. M. I. Taha and P. Schaumont. Side-Channel Analysis of MAC-Keccak. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013, Austin, TX, USA, June 2-3, 2013*, pages 125–130, 2013.

[28] C. D. Walter. Sliding Windows Succumbs to Big Mac Attack. In c. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 286–299. Springer Berlin Heidelberg, 2001.