A Horizontal Attack on the Codes and Restricted Objects Signature Scheme (CROSS)

Jonas Schupp[1] $^{[0000-0002-4171-1656]}$ and Georg Sigl[1,2] $^{[0000-0003-3152-941X]}$

[1] TUM School of Computation, Information and Technology
Technical University of Munich, Munich, Germany
`jonas.schupp,sigl@tum.de`

[2] Fraunhofer Institute for Applied and Integrated Security (AISEC), Garching, Germany

Abstract. CROSS is a post-quantum secure digital signature scheme submitted to NIST's Call for Additional Signatures which was recently selected for round 2. It features signature and key sizes in the range of SLH-DSA while providing a substantially faster signing operation. Within this work, we provide the first passive side-channel attack on the scheme. The attack recovers the secret key from all except one parameter sets and security levels from a single power trace while requiring at maximum two power traces for the R-SDP(G) 1 Fast instance. To successfully mount the attack, we show how to recover the secret key from side-channel information gained from the syndrome computation in CROSS' identification protocol. We furthermore show how the hypothesis space for the attack can be restricted using information from the published signature.

Keywords: Post-Quantum Cryptography· Side-Channel· Horizontal Attack

1 Introduction

Given the possible insecurity of classic asymmetric cryptography in the presence of a quantum computer being capable of running Shor's Algorithm [10], the National Institute of Standardization (NIST) started a competition to find quantum secure replacements in 2016 [7]. In July 2022, NIST selected the algorithms Crystals-KYBER as Key Encapsulation Mechanism as well as Crystals-Dilithium and Falcon as signature algorithms [2]. All of these algorithms, apart from the already selected SPHINCS+/SLH-DSA [1] and a yet to be chosen Key Encapsulation Mechanism, are based on structured lattices. This led to an additional call by NIST asking specifically for signature schemes based on different mathematical problems. There are several advantages of having schemes based on different problems available. On one hand, this allows for alternatives in case there is a breakthrough in cryptanalysis for one of the problems being used and on the other hand it allows for schemes with specific advantages, e.g. small signatures or low computational requirements to be used for specific applications.

Of the 40 submissions accepted by NIST in June 2023, they selected 14 to advance to round 2 [3]. One of these is the Codes and Restricted Objects Signature Scheme (CROSS). It is a promising candidate as it offers reasonable signature and key sizes which are in the range of SPHINCS+ but offers significantly faster signing and key generation routines according to [3].

NIST also asked for research in side-channel resistance of the new schemes as side-channel attacks are an eminent threat for cryptographic implementations, especially on embedded devices. As there is to the best of our knowledge no research on the passive side-channel resistance of CROSS so far, we propose the first horizontal attack on this signature scheme requiring only a single power trace for all except one of the parameter sets and at maximum two traces for the R-SDP(G) 1 Fast parameter set.

Related Work While we are not aware of any passive attacks published on CROSS, the attack strategy we follow is similar to the one introduced by Clavier et al. in [8] and used by Bauer et al. in [5] for ECC implementations. Their work was furthermore inspired by the work of Walter in [11]. We furthermore rely on the concept of Correlation Power Analysis (CPA) as introduced by Brier et al. in [6] and use Hamming Weights as our leakage model.

Contribution In this work, we present the first power side-channel attack on CROSS which requires a trace of a single signature generation for all except one of the parameter sets and at maximum two traces for the R-SDP(G) 1 Fast parameter set. The attack works with the outlined performance against all security levels and optimization corners. Our target is the latest reference implementation of the scheme as available on the submission team's webpage.[3] Please note that while the attack targets the latest version available for Round 1 of the NIST competition (Version 1.2) it is directly and without modification applicable to the Round 2 version of the scheme. The only thing to be taken into consideration is the altered notation. The attack leads to full recovery of a representation of the secret key which can be used to sign arbitrary messages. Our attack target is the syndrome computation in each round of the identification protocol underlying CROSS which is essentially a matrix-vector multiplication. Instead of computing the full matrix vector product, part of the multiplication where the upper vector entries are multiplied with an identity matrix is implemented by priming the result vector with these values. This forces us to implement our attack in several stages as we can build hypothesis on the upper part of the vector before recovering the lower part. First, we target the intermediate results of the multiplication with a horizontal correlation based attack which allows us to recover most of the respective coefficients under attack. Second, we target the intermediate results of the addition of the multiplication results which we can now compute based on the first attack's result to attack the remaining coefficients. Additionally, we outline how to recover the secret key of the scheme

[3] https://www.cross-crypto.com/CROSS_submission_package_v1.2.zip

from the ephemeral secret sampled in each execution of the identification scheme using the information recovered with our attack. We also show how to recombine the information gained in several rounds of the protocol. Furthermore, we explain how to reduce the number of required key hypothesis based on the special structure of the restricted syndrome decoding problem. Finally, we show the practicality of our attack with traces measured on ChipWhisperer's CW308 platform using an STM32F303 microcontroller.

Organization The remainder of this work is structured as follows: Within chapter two, we outline CROSS with emphasis on the parts which are necessary for the attack. In chapter three, we explain the necessary stages to mount a successful attack on the algorithm. Finally, in chapter four, we show experimental results from measurements on a microcontroller and demonstrate the feasibility of the attack.

2 Introduction to CROSS

In the following, we provide an overview of CROSS according to the latest version of its specification [4]. The scheme is based on the hardness of the Restricted Syndrome Decoding Problem (R-SDP) which can be seen as a variant of the syndrome decoding problem. Additionally, it features a second variant based on the Restricted Syndrome Decoding Problem in a Subgroup (R-SDP(G)). It is constructed using the Fiat-Shamir transform to make an interactive Zero-Knowledge Identification protocol non interactive. The scheme has relatively small signatures and keys, at least compared to other code-based schemes, while showing competitive performance when it comes to computational complexity. On a high level perspective, the scheme is constructed from a Zero-Knowledge Identification protocol which is executed for t rounds and explained in the following section.

Table 1 shows the parameters for all instantiations of the scheme with p and z being the moduli for the respective finite fields the coefficients in the scheme are sampled from. The codewords of the random linear code used in the scheme have length n and dimension k while the number of rounds is defined as t and the fixed weight of the second challenge is w. In the next sections, we provide a high level overview of the main functions of the scheme while omitting details which are not relevant for this work.

Notation We here define the notation we use throughout this work. For any uncovered notation we refer to [4] as we omit any notation used in the pseudocode of the signing operation in case it is irrelevant for the attack. Matrices are represented by uppercase letters $\mathbf{A}$ while vectors are indicated as lowercase letters $\mathbf{v}$. The identity matrix of size m is denoted as $\mathbf{I}_m$. Any representations of length n vectors in $\mathbb{F}_z$ are denoted by lowercase bold Greek letters, e.g. $\boldsymbol{\eta}$.

Table 1: Parameter choices, keypair and signature sizes recommended for both CROSS-R-SDP and CROSS-R-SDP(G), assuming NIST security categories 1, 3, and 5, respectively.

Algorithm and Security Category	**Optim. Corner**	p	z	n	k	m	t	w	**Pri. Key Size (B)**	**Pub. Key Size (B)**	**Signature Size (B)**
CROSS-R-SDP **1**	**fast**	127	7	127	76	-	163	85	32	77	19152
	balanced	127	7	127	76	-	252	212	32	77	12912
	small	127	7	127	76	-	960	938	32	77	10080
CROSS-R-SDP **3**	**fast**	127	7	187	111	-	245	127	48	115	42682
	balanced	127	7	187	111	-	398	340	48	115	28222
	small	127	7	187	111	-	945	907	48	115	23642
CROSS-R-SDP **5**	**fast**	127	7	251	150	-	327	169	64	153	76298
	balanced	127	7	251	150	-	507	427	64	153	51056
	small	127	7	251	150	-	968	912	64	153	43592
CROSS-R-SDP(G) **1**	**fast**	509	127	55	36	25	153	79	32	54	12472
	balanced	509	127	55	36	25	243	206	32	54	9236
	small	509	127	55	36	25	871	850	32	54	7956
CROSS-R-SDP(G) **3**	**fast**	509	127	79	48	40	230	123	48	83	27404
	balanced	509	127	79	48	40	255	176	48	83	23380
	small	509	127	79	48	40	949	914	48	83	18188
CROSS-R-SDP(G) **5**	**fast**	509	127	106	69	48	306	157	64	106	48938
	balanced	509	127	106	69	48	356	257	64	106	40134
	small	509	127	106	69	48	996	945	64	106	32742

2.1 The CROSS-ID protocol

The Zero-Knowledge protocol used in CROSS is built upon a 5-pass protocol, typically consisting of a commitment sent by the prover to the verifier, and a pair of a challenge and a response sent from the verifier to the prover and back which is repeated twice with different messages. Depending on the values of the two challenges, the prover provides different responses in order to prove its knowledge of the secret error vector $\boldsymbol{\eta}$ (resp. $\boldsymbol{\zeta}$ for R-SDP(G)) without actually revealing it. Within this work, we however only focus on the computation of the commitment which depends on the secret key and on a subset of rounds, namely on those where second binary challenge is 1. In order to compute the commitments one first needs to sample an ephemeral secret $\boldsymbol{\eta}'$ as well as a vector $\mathbf{u}'$ uniform in $\mathbb{F}_p$. These can then be used to compute the transformation σ in $\mathbb{F}_z$ which maps $\boldsymbol{\eta}'$ to $\boldsymbol{\eta}$ and can similarly be used to map $\mathbf{u}'$ to $\mathbf{u}$. With $\mathbf{u}$, one can then compute the syndrome $\tilde{\mathbf{s}}$. Next, a hash of the syndrome $\tilde{\mathbf{s}}$ and the map σ on one hand as well as of the Seed used to sample $\boldsymbol{\eta}'$ and $\mathbf{u}'$ on the other hand is published. Depending on the challenges provided by the verifier one then computes a response based on the syndrome and the first challenge and finally reveals either the seed used to sample $\boldsymbol{\eta}'$ and $\mathbf{u}'$ in case the second binary challenge is 1 or a computed value $\mathbf{y}$ and the map σ which maps $\boldsymbol{\eta}'$ to $\boldsymbol{\eta}$ and $\mathbf{u}'$ to $\mathbf{u}$.

2.2 Key Generation

The key generation has a keypair consisting of a public and a secret key as output. The secret key consists of a seed used to sample the error vector $\boldsymbol{\eta}$ (resp. $\boldsymbol{\zeta}$). The public key contains a seed used to sample the parity check matrix $\mathbf{H}$ as well as the syndrome $\mathbf{s}$ of $\boldsymbol{\eta}$ computed through $\mathbf{H}$. To reduce the key sizes, neither $\boldsymbol{\eta}$ nor $\mathbf{H}$ are saved directly but both are only stored as their respective seeds. For R-SDP(G) the error vector $\boldsymbol{\eta}$ is computed from $\boldsymbol{\zeta}$ using the generator matrix $\mathbf{M}_G$ which is also part of the public key. We refer to [4] for a more detailed description as well as for the corresponding pseudocode.

2.3 Signature Generation

Within the signature generation algorithm, the secret key and the message to be signed are taken as input to compute the respective signature. Please note that the public key can be recomputed from the secret key as the public key's seed is derived from the private key's seed. After expanding the matrix $\mathbf{H}$ as well as the error vector $\boldsymbol{\eta}$, a seed and a salt is drawn from the system's TRNG and expanded for each round using a binary tree structure. These seeds are then used in the CROSS-ID protocol described above and finally the responses to be published are packed to form the actual signature. The pseudocode of the signature generation is shown in Algorithm 1 where line 2 refers to the expansion of the key material, the commitments are computed and packed in lines 3-17 and the challenges and responses are calculated in 18-31. The signature is finally packed in lines 32-42. Within this algorithm orange highlights operations carried out only for the RSDP-(G) parameter sets while green is used for parts only computed for RSDP.

2.4 Signature Verification

In the signature verification, one of the two following actions is executed based on the corresponding value of the second challenge. Either the values $\boldsymbol{\eta}'$ and $\mathbf{u}'$ are sampled based on the published seed or the map σ is used to recompute $\tilde{\mathbf{s}}$ in combination with $\mathbf{y}$. Finally it is checked whether these values match the ones computed during signature generation via hash comparison and subsequently a decision about the correctness of the signature is made.

3 Attack Strategy

As discussed above, the secret key of CROSS consists only of a seed of 2λ length. Within this work we do not target that seed directly but instead the error vector $\boldsymbol{\eta}$ (resp. $\boldsymbol{\zeta}$ in the case of R-SDP$((G))$) which is the only secret material sampled from the seed and therefore corresponds to full secret key recovery. As visible in Algorithm 1, the only operation in the algorithm where $\boldsymbol{\eta}$ is used directly is to compute the map σ in line 7. This line represents the transformation between the longterm secret $\boldsymbol{\eta}$ and the ephemeral secret $\boldsymbol{\eta}'$ which is sampled uniform at

Algorithm 1: $\textsc{Sign}(\texttt{pri}, \texttt{Msg})$, taken from [4]

```
Data: $\lambda$: security parameter,
      $g \in \mathbb{F}_p^*$: generator of a subgroup $\mathbb{E}$ of $\mathbb{F}_p^*$ with cardinality $z$
      $\mathbb{E}^n$: restricted subgroup
      $\mathbf{M}_G$: $m \times n$ matrix of $\mathbb{Z}_z$ elements, employed to generate vectors
      $\boldsymbol{\eta} \in G \subset \mathbb{E}^n$
      $t$: number of iterations of the ZKID protocol
      $\mathcal{B}_w^t$: set of all binary strings with length $w$ and Hamming weight $t$
      $c$: a fixed constant, equal to the number of nodes in the seed tree
      $\texttt{dsc}$: a fixed constant, greater than $t$ employed to obtain domain
      separation
Input: $\texttt{pri}$: private key constituted of $\texttt{Seed}_{\texttt{sk}} \in \{0,1\}^{2\lambda}$
       $\texttt{Msg}$: message to be signed $\texttt{Msg} \in \{0,1\}^*$
Output: Signature $\texttt{Signature}$
Begin
    // Key material expansion
    $\boldsymbol{\eta}, \boldsymbol{\zeta}, \mathbf{H}, \mathbf{M}_G \leftarrow \textsc{ExpandPrivateSeed}(\texttt{Seed}_{\texttt{sk}})$
    $\boldsymbol{\eta}, \mathbf{H} \leftarrow \textsc{ExpandPrivateSeed}(\texttt{Seed}_{\texttt{sk}})$
    // Computation of commitments
    $\texttt{Mseed} \xleftarrow{\$} \{0,1\}^{\lambda}$, $\texttt{Salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$
    $(\texttt{Seed}[0], \ldots, \texttt{Seed}[t-1]) \leftarrow \textsc{SeedTreeLeaves}(\texttt{Mseed}, \texttt{Salt})$
    for $i \leftarrow 0$ to $t-1$ do
        $\boldsymbol{\zeta}', \boldsymbol{u}_i' \leftarrow \text{CSPRNG}\ (\texttt{Seed}[i] || \texttt{Salt} || i + c, )$
        $\boldsymbol{\delta}_i \leftarrow \boldsymbol{\zeta} - \boldsymbol{\zeta}'$
        $\boldsymbol{\eta}_i' \leftarrow \boldsymbol{\zeta}' \mathbf{M}_G$
        $\boldsymbol{\eta}_i', \boldsymbol{u}_i' \leftarrow \text{CSPRNG}\ (\texttt{Seed}[i] || \texttt{Salt} || i + c, )$
        $\boldsymbol{\sigma}_i \leftarrow \boldsymbol{\eta} - \boldsymbol{\eta}_i'$
        for $j \leftarrow 0$ to $n-1$ do
            $\boldsymbol{v}[j] \leftarrow g^{\boldsymbol{\sigma}_i[j]}$
        end
        $\boldsymbol{u} \leftarrow \boldsymbol{v} \star \boldsymbol{u}_i'$ // $\star$ is component-wise product
        $\tilde{\mathsf{s}} \leftarrow \boldsymbol{u}\mathbf{H}^{\top}$
        $\mathsf{cmt}_0[i] \leftarrow \textsc{Hash}(\tilde{\mathsf{s}} || \boldsymbol{\delta}_i || \texttt{Salt} || i + c + \texttt{dsc})$
        $\mathsf{cmt}_0[i] \leftarrow \textsc{Hash}(\tilde{\mathsf{s}} || \boldsymbol{\sigma}_i || \texttt{Salt} || i + c + \texttt{dsc})$
        $\mathsf{cmt}_1[i] \leftarrow \textsc{Hash}(\texttt{Seed}[i] || \texttt{Salt} || i + c + \texttt{dsc})$
    end
    $\mathsf{d}_0 \leftarrow \textsc{MerkleRoot}(\mathsf{cmt}_0[0], \ldots, \mathsf{cmt}_0[t-1])$
    $\mathsf{d}_1 \leftarrow \textsc{Hash}(\mathsf{cmt}_1[0]\ ||\ \ldots\ ||\ \mathsf{cmt}_1[t-1])$
    $\mathsf{d}_{01} \leftarrow \textsc{Hash}(\mathsf{d}_0\ ||\ \mathsf{d}_1)$

    // First challenge vector extraction
    $\mathsf{d}_m \leftarrow \textsc{Hash}(\mathsf{Msg})$
    $\mathsf{d}_\beta \leftarrow \textsc{Hash}(\mathsf{d}_m || \mathsf{d}_{01} || \texttt{Salt})$
    $\mathsf{beta} \leftarrow \mathrm{CSPRNG}\left(\mathsf{d}_\beta, (\mathbb{F}_p^*)^t\right)$
    // Computation of first round of responses
    for $i \leftarrow 0$ to $t-1$ do
        for $j \leftarrow 0$ to $n-1$ do
            $\mathbf{e}'_i[j] \leftarrow g^{\boldsymbol{\eta}'_i[j]}$
        end
        $\mathbf{y}_i \leftarrow \boldsymbol{u}'_i + \mathsf{beta}[i]\mathbf{e}'_i$
    end
    // Second challenge vector extraction
    $\mathsf{d}_b \leftarrow \textsc{Hash}(\mathbf{y}_0 || \ldots || \mathbf{y}_{t-1} || \mathsf{d}_\beta)$
    $\mathbf{b} \leftarrow \mathrm{CSPRNG}\left(\mathsf{d}_b, \mathcal{B}^t_{(w)}\right)$
    // Computation of second round of responses
    $\mathsf{MerkleProofs} \leftarrow \textsc{MerkleProof}((\mathsf{cmt}_0[0], \ldots, \mathsf{cmt}_0[t-1]), \mathbf{b})$
    $\mathsf{SeedPath} \leftarrow \textsc{SeedTreePaths}(\texttt{Mseed}, \mathbf{b})$
    // Signature composition
    $\mathsf{rsp}_0 \leftarrow (\mathbb{F}_p^n \times \mathbb{F}_z^m)^{t-w}$ | $\mathsf{rsp}_0 \leftarrow (\mathbb{F}_p^n \times \mathbb{F}_z^n)^{t-w}$
    $\mathsf{rsp}_1 \leftarrow (\{0,1\}^{2\lambda})^{t-w}$ // empty array
    $j \leftarrow 0$
    for $i \leftarrow 0$ to $t-1$ do
        if $(\mathbf{b}[i] = 0)$ then
            // $\mathsf{cmt}_0[i]$ is recomputed by the verifier, $\mathsf{cmt}_1[i]$ must be
             sent
            $\mathsf{rsp}_0[j] \leftarrow (\mathbf{y}_i, \boldsymbol{\delta}_i)$ | $\mathsf{rsp}_0[j] \leftarrow (\mathbf{y}_i, \boldsymbol{\sigma}_i)$
            $\mathsf{rsp}_1[j] \leftarrow \mathsf{cmt}_1[i]$
            $j \leftarrow j + 1$
        end
    end
    $\texttt{Signature} \leftarrow \texttt{Salt} \,||\, \mathsf{d}_{01} \,||\, \mathsf{d}_b \,||\, \mathsf{MerkleProofs} \,||\, \mathsf{SeedPath} \,||\, \mathsf{rsp}_0 \,||\, \mathsf{rsp}_1$
    // all Signature components are encoded as binary strings
    return Signature
end
```

random in $\mathbb{F}_z^n$ for every round of the zero-knowledge protocol. Depending on the corresponding bit of the second challenge, either the seed used to sample $\boldsymbol{\eta}'$ (and $\mathbf{u}'$) or the map σ is published. For this attack, we focus on the w of t fraction of rounds where the seed gets published and we can therefore obtain $\mathbf{u}'$ and $\boldsymbol{\eta}'$. One can furthermore notice that the correspondence between $\mathbf{u}$ and $\mathbf{u}'$ is derived from the same map as the transformation between $\boldsymbol{\eta}$ and $\boldsymbol{\eta}'$. A recovery of a pair of $\mathbf{u}$ and $\mathbf{u}'$ can therefore be used to recover $\boldsymbol{\eta}$ from $\boldsymbol{\eta}'$.

Recovering $\boldsymbol{\eta_i}$ from a pair of values $\boldsymbol{u_i}$ and $\boldsymbol{u_i'}$ Given a pair of u_i and u_i' this can be achieved by recovering the map v_i from the pair, looking up the precomputed value of σ_i for the corresponding entry in $\mathbf{v}$ based on a Look-Up Table and recomputing η_i from the map σ_i and η_i' as visible on the following lines.

$$\begin{aligned}
&v_i = (u_i \cdot u_i'^{-1}) \mod p\\
&\sigma_i \Leftarrow v_i\\
&\eta_i = (\eta_i' \cdot \sigma_i) \mod z
\end{aligned}$$

Component-wise transformation It is furthermore worth noticing that the transformation σ and therefore also v is defined component-wise, allowing us to recover $\boldsymbol{\eta}$ element by element from any pair of $\mathbf{u}$ and $\mathbf{u}'$. This allows us to gain information on a limited number of coefficients of $\mathbf{u}$ per round and to combine this information later on to learn the longterm secret $\boldsymbol{\eta}$. In the case of R-SDP(G) we can then use $\boldsymbol{\eta}$ to recover $\boldsymbol{\zeta}$ directly as $\boldsymbol{\eta} \leftarrow \boldsymbol{\zeta}\mathbf{M}_G$ which allows us to recover $\boldsymbol{\zeta}$ directly as part of $\boldsymbol{\eta}$.

$$\boldsymbol{\zeta} = [\eta_{n-m}; ...; \eta_n]$$

Reducing the hypothesis space Another aspect that simplifies the attack here is the way the coefficients of $\mathbf{u}$ are computed. We recall here that $\mathbf{u}'$ is sampled uniform at random from $\mathbb{F}_p^n$ while the map $\mathbf{v}$ is a representation of the map $\sigma \in \mathbb{F}_z^n$. Therefore σ can take only $|z|$ possible values for each entry which leads to the same limitation for $\mathbf{v}$ and thus requires us to consider only $|z|$ hypotheses for each entry in $\mathbf{u}$. These can be computed by taking all possible values of each entry in $\mathbf{v}$ into account and computing the resulting values $\mathbf{u}$ given that $\mathbf{u}'$ is published as part of the signature.

With this approach, we only need an exploitable operation in the code to recover a coefficient of $\mathbf{u}$ in a round where $\mathbf{u}'$ is published as part of the signature. Such a spot exists via the syndrome computation where $\mathbf{u}$ gets multiplied with the parity check matrix $\mathbf{H}$ which is part of the public key to obtain the syndrome y. While the (restricted) syndrome decoding problem itself is computationally hard, its intermediate results can still be used in an implementation attack in case they are accessible via a side-channel. Within the next sections, we outline how we can extract the entire secret key representation $\boldsymbol{\eta}$ from this syndrome computation using only a single trace for all parameter sets except one and at maximum two traces for R-SDP(G) 1 Fast.

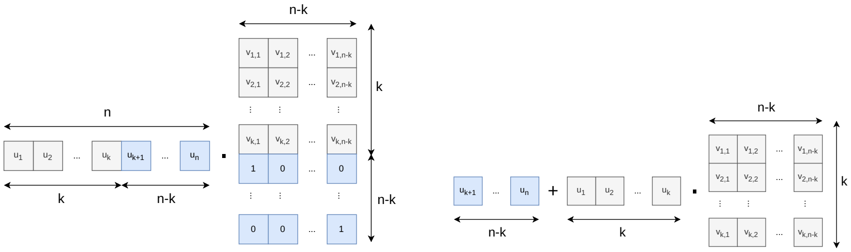
n-k
$v_{1,1}$ $v_{1,2}$ ... $v_{1,n-k}$
$v_{2,1}$ $v_{2,2}$ ... $v_{2,n-k}$
k
n
u_1 u_2 ... u_k u_{k+1} ... u_n
$v_{k,1}$ $v_{k,2}$ ... $v_{k,n-k}$
1 0 ... 0
0 0 ... 1
k
n-k
n-k
n-k
u_{k+1} ... u_n + u_1 u_2 ... u_k
$v_{1,1}$ $v_{1,2}$ ... $v_{1,n-k}$
$v_{2,1}$ $v_{2,2}$ ... $v_{2,n-k}$
$v_{k,1}$ $v_{k,2}$ ... $v_{k,n-k}$
k
n-k
k

Fig. 1: Illustration of the matrix-vector multiplication used in the syndrome computation and how the multiplication with the identity matrix is substituted

3.1 Prerequisites

The operation we target with our attack is essentially a matrix vector multiplication in $\mathbb{F}_p$ with $p = 127$ for R-SDP and $p = 509$ for R-SDP(G). While the operands in this multiplication are actually of that size and bitwidth, we want to recall here that our hypothesis space is limited by the way the values of the vector are sampled which leaves us only $z = 7$ hypotheses per entry in $\mathbf{u}$ for the R-SDP parameter sets and $z = 127$ hypotheses per entry for the R-SDP(G) instances. As the systematic part of the parity check matrix $\mathbf{H}$ is just an identity matrix of dimension $(n-k) \times (n-k)$, this part of the multiplication is usually omitted and replaced by copying the corresponding $(n-k)$ values of $\mathbf{u}$ as initialization values into the result vector $\tilde{\mathbf{s}}$ before the multiplication results are accumulated as sketched in Equation (1). This leaves us with the pseudocode as shown in Algorithm 2 which we also visualize in Figure 1. Here one can see that after copying the upper part of $\mathbf{u}$ to $\tilde{\mathbf{s}}$ we are left with $(n-k) \times k$ multiplications, subsequent additions to the corresponding entry in $\tilde{\mathbf{s}}$ and finally reductions which form the entire matrix vector multiplication. To mount our attack, we need to capture each of these operations for a single matrix-vector multiplication as the vector $\mathbf{u}$ changes for each multiplication.

$$\tilde{\mathbf{s}} = \mathbf{u}\mathbf{H}^{\top} \tag{1}$$

$$\tilde{\mathbf{s}} = \mathbf{u}\left[\mathbf{V}_{tr}\mathbf{I}_{n-k}\right]^{\top} \tag{2}$$

$$(s_1, ..., s_{n-k}) = (u_1, ..., u_n)\begin{pmatrix} v_{tr,1,1} & ... & v_{tr,1,n-k} \\ v_{tr,2,1} & ... & v_{tr,2,n-k} \\ ... & ... & ... \\ v_{tr,k,1} & ... & v_{tr,k,n-k} \\ 1 & ... & 0 \\ 0 & ... & 0 \\ ... & ... & ... \\ 0 & ... & 1 \end{pmatrix} \tag{3}$$

$$(s_1, ..., s_{n-k}) = (u_{k+1}, ..., u_n) + (u_1, ..., u_k)\begin{pmatrix} v_{tr,1,1} & ... & v_{tr,1,n-k} \\ v_{tr,2,1} & ... & v_{tr,2,n-k} \\ ... & ... & ... \\ v_{tr,k,1} & ... & v_{tr,k,n-k} \end{pmatrix} \tag{4}$$

Algorithm 2: Syndrome Computation

```
1 for $i \leftarrow 1$ to $n-k$ do
2     $\tilde{s}_i = u_{k+i}$
3 end
4 for $i \leftarrow 1$ to $k$ do
5     for $j \leftarrow 1$ to $n-k$ do
6         $s_i = (s_i + u_i \cdot v_{tr,i,j}) \mod p$
7     end
8 end
```

3.2 CPA on the Syndrome Computation to Recover the Lower k Entries of $\mathbf{u}$

To mount the attack, we can now compute hypotheses on the intermediate results of the multiplications in line 6 of Algorithm 2 and run a CPA-style attack on the corresponding $n-k$ samples per coefficient in $\mathbf{u}$. We illustrate this attack strategy in Figure 2 where we try to attack the red value via the intermediate results of its multiplications with the yellow elements of the public matrix. As discussed above, this leaves us with z possible values for an entry in $\mathbf{u}$ and, as $\mathbf{V}_{tr}$ is part of the public key, we can compute hypotheses on the (unreduced) multiplication results using a Hamming Weight Leakage Model. The necessary samples here all belong to one iteration of the outer loop in Algorithm 2 and consist of one full execution of the inner loop, meaning that we need the first $n-k$ subtraces to attack the first coefficient in $\boldsymbol{u}$. Please note that the number

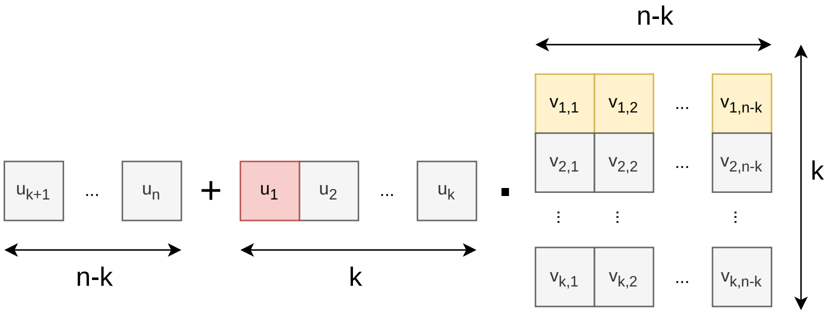

Fig. 2: Illustration of the attack path for the CPA-Style attack on the lower k entries of $\mathbf{u}$ resp. $\boldsymbol{\eta}$

of attack points here is relatively limited as e.g. for RSDP-1 we only have 51 samples per coefficient and e.g. for RSDP(G)-1 we even only have 19 samples. Under the assumption of Hamming Weight Leakage, we can not expect to obtain unique results per coefficient here as there is usually more than one possible hypothesis leading to the same sequence of hamming weights. However, we can discard non-unique samples or putting it differently, samples where more than one coefficient correlates within a certain threshold of the most likely hypothesis. As we also got more than one chance to recover each entry in $\mathbf{u}$ as there are w (which is at minimum 79 for RSDP(G)-1) rounds where $\mathbf{u}'$ gets published, we can safely discard these ambiguous results and can still recover most of the entries in $\mathbf{u}$ successfully as shown below. As we essentially have w attack results per coefficient in $\mathbf{u}$, though we discard those which are not distinct, we also apply a majority voting based strategy to further reduce the number of incorrectly recovered coefficients. To summarize this leaves us with a horizontal DPA for each of the k first coefficients of $\boldsymbol{u}$ for w rounds. In the case of R-SDP, a recovery of the k entries in different vectors $\mathbf{u}$ equal to a recovery of this part of the secret key error vector $\boldsymbol{\eta}$. In the case of R-SDP(G), we need the upper m entries of $\boldsymbol{\eta}$ which form the secret key $\boldsymbol{\zeta}$. This attack result is nevertheless necessary as it is prerequisite to attack the upper part of $\boldsymbol{\eta}$.

3.3 CPA on the Syndrome Computation to Recover the Upper $n - k$ Entries of u

In order to successfully attack the second part of the coefficients in $\mathbf{u}$ one first needs to recover a significant part of $\mathbf{u}$ successfully using the strategy described in Section 3.2. Once this is done, one can now compute new hypotheses along the other axis of the parity check matrix respectively the calculations corresponding to this dimension. We illustrate our approach in Figure 3 where we attack the red value via hypotheses we compute on the reduced addition results. We can

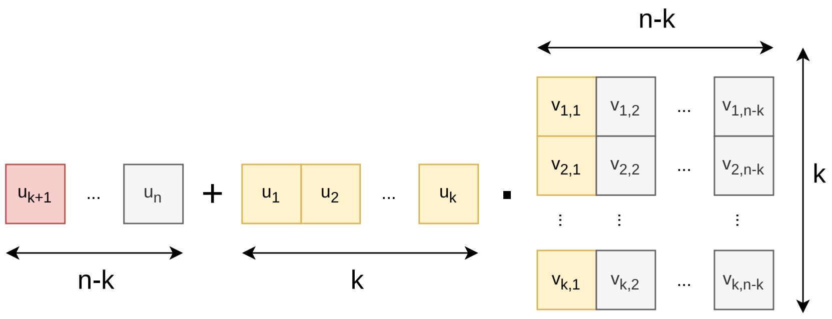

Fig. 3: Illustration of the attack path for the CPA-Style attack on the upper $n-k$ entries of $\mathbf{u}$ resp. $\boldsymbol{\eta}$

compute these hypotheses because we can calculate the multiplication results of the yellow column of the matrix with the vector $\mathbf{u}$ based on the results of the first attack stage. Putting it differently, to attack the entry u_{k+1} we need the traces $1, (n-k)+1, 2 \cdot (n-k)+1, ..., k*(n-k)+1$. This yields k subtraces per coefficient to be attacked in $\boldsymbol{u}$ and therefore, as $k > (n-k)$ for all parameter sets, a more reliable attack, simply due to the larger number traces available. As we compute our hypotheses here based on the results of the addition as described in line 6 Algorithm 2, we need to know the first k entries in $\boldsymbol{u}$ to precompute the results of the respective multiplications successfully such that the only unknown input is the entry of $\boldsymbol{u}$ copied into the respective position in $\tilde{\mathbf{s}}$ in the first place and accumulated in all subsequent iterations.

4 Results

To assess the feasibility and efficiency of the attacks described above, we first evaluate them on simulated traces before performing the attack on an actual implementation on an ARM Cortex-M4 based microcontroller as described below. Please note that throughout the results section, we usually talk about recovery of values in $\boldsymbol{\eta}$ which always refer to an attack on the corresponding value in $\mathbf{u}$ and subsequent recovery via $\mathbf{u}'$ and $\boldsymbol{\eta}'$.

Results based on simulated Hamming Weight Leakages In this stage, we compute upon the following expectations which can be assumed to be applicable to a microcontroller implementation in general. First, we expect noisy Hamming Weight Leakage from all results which are written back into a register at some point during the execution of the algorithm. This is e.g. experimentally confirmed for load and store operations in [9]. As we first only want to assess the theoretical feasibility of the attack, we assume this leakage to be noise free as we try to reproduce our results with real measurements anyways. More concretely, we therefore expect leakage from the following (intermediate) results of the syndrome computation:

- The unreduced result of one multiplication $u_i \cdot v_{tr,i,j}$
- The reduced result of one addition $(s_i + u_i \cdot v_{tr,i,j}) \mod p$

The artificial traces we build from these expectation consist of two sample points per atomic operation, one being the Hamming Weight of the unreduced result of one multiplication as outlined above and one being the Hamming Weight of the reduced addition result. While this model is an oversimplification it is device independent and sufficient to assess whether an attack is feasible in an ideal scenario with the leakages described above. We discuss below how well these expectations match reality, for now, we assume them to correct.

Results based on measurements We perform our attack on a ChipWhisperer CW308 board as platform with an 32-bit STM32F303RCT7 microcontroller running at 10 MHz. As measurement oscilloscope we use a Picoscope 6402D with a sampling rate of 2.5 Gs/s measuring the power consumption of our device via a shunt resistor and a DC-Block with 20 MHz cutoff frequency. We use `arm-none-eabi-gcc` in version 13.2.1 for all experiments in this work. On the implementation side, we had to make some adaptions to the reference implementation due to the limited SRAM size of the victim board. Our targeted implementation therefore currently only contains the side-channel relevant functions as well as the functions necessary to compute its inputs. Regarding the expected leakage points above, we can confirm that we can exploit leakage from all mentioned intermediate results though the practicability heavily depends on the chosen parameter set. Concretely, there is less leakage for R-SDP(G) which we mainly attribute to the lower number of subtraces per attack and the larger coefficients as $p = 509$. To get distinct results for R-SDP(G) we therefore also have to narrow down which part of the trace we were actually attacking. For R-SDP we take the trace of all operations within a loop iteration while for R-SDP(G) we undertake some further refinement steps. Here, we use a Signal-to-Noise calculation to determine where information about the public matrix entries is visible in the trace. In a second step, we then only use the intervals following the points where information about these entries is present as the public matrix entries are one operand of the multiplication, indicating that the desired value should be computed at some point afterwards.

4.1 Result of a single CPA

The smallest unit of the attack strategy described above is one CPA with a Hamming Weight based hypothesis on either the multiplication or subsequent addition result along either a row or a column of the public matrix with an element of the vector under attack. In the simulated setting, we get distinct results in between 20% and 70% of the attacks depending on the parameter set.

We show a sample of the to be expected correlation coefficients in Figure 4 for R-SDP 1 Fast as well as in Figure 5 for R-SDP(G) 1 Fast. Both plots show an example for a coefficient where we get a distinct result, i.e. there is exactly one maximum correlation, as well as a sample where more than one hypothesis yields a similarly high correlation, as mentioned above, we discard such samples. We furthermore choose a threshold of 80% of the maximum correlation which needs to be the maximum magnitude for the second highest value to consider the highest one distinct. This threshold is visualized by the red lines in the above mentioned figures. While this threshold was initially chosen based on simulated results, we observed that it seems to also work well for real traces of the R-SDP parameter sets. For the R-SDP(G) parameter sets, we had to increase the threshold to 95% for the first attack stage described below as the limited number of attack traces per CPA does not yield such high differences in the correlation coefficient. Please note that this approach discards several correct but not sufficiently distinct results but leaves us with an acceptable ratio of correct choices for the subsequent attack steps.

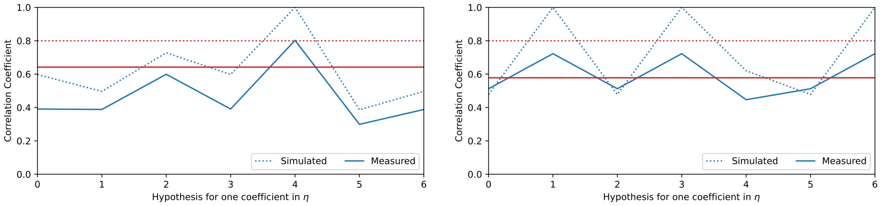

Fig. 4: Simulated and measured correlation coefficient for two coefficients in $\boldsymbol{\eta}$ for the R-SDP 1 Fast instance of which one yields a distinct and one a not distinct result. The decision boundary of 80% of the maximum magnitude is marked as red line

4.2 Attack on the first k entries of $\boldsymbol{\eta}$

After obtaining distinct results of several CPAs for each of the lower k entries of $\boldsymbol{\eta}$ from several rounds of CROSS' identification protocol, we now employ a majority voting based strategy the recover a single coefficient from a set of distinct CPA results. Here we first calculate the corresponding entry in $\boldsymbol{\eta}$ for a result as discussed in Section 3 and then accumulate these results for each possible value for this entry in $\boldsymbol{\eta}$ which leaves us with a distinct result for this entry. We also choose here to only consider results which are at least recovered twice to further reduce the number of errors in the result. To visualize this approach, we plotted the number of recovered entries in $\boldsymbol{\eta}$ as well as the number of distinct CPA results against the number of attacked rounds from a single signature generation in Figure 6 for R-SDP 1 Fast and in Figure 7 for R-SDP(G) 1 Fast. As

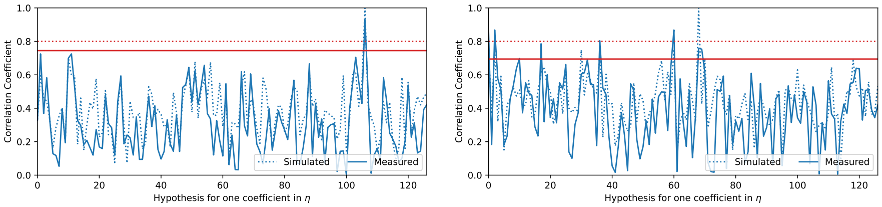

Fig. 5: Simulated and measured correlation coefficient for two coefficients in $\boldsymbol{\eta}$ for the R-SDP(G) 1 Fast instance of which one yields a distinct and one a not distinct result. The decision boundary of 80% of the maximum magnitude is marked as red line

visible in Figure 6, we recover all k lower entries for R-SDP 1 Fast within less than 30 attackable rounds thus leaving 55 rounds headroom to either decrease the necessary measurement length within one signature generation or to recover results in a noisier setting from a single signing operation. We furthermore want to point out that the attacks work despite a significant number of incorrect distinct results which are indicated by the dashed lines. In the case of R-SDP(G) 1 Fast, as shown in Figure 7 we fail to recover all lower k values in $\boldsymbol{\eta}$ in the measured setting while this should in theory be feasible as illustrated by the dotted red line reaching $k = 36$ within the rounds contained in one signature. This result is partially due to the fact that we only have 19 subtraces per CPA as well as larger values as $p = 509$ in this case. Here we need additional results from further signing operations to successfully recover all lower k entries. To give an overview over all results for this first attack stage, we refer to Table 2 where we summarize the number of distinct hypothesis we gain per round of the ID-Protocol, the fraction of correct hypothesis of those, as well as the resulting number of rounds of the ID-Protocol required to recover the lower k entries of $\boldsymbol{\eta}$.

4.3 Attack on the upper $n - k$ entries of η

As discussed in Section 3, we rely on the lower k entries of $\boldsymbol{\eta}$ to successfully compute hypotheses on the upper part, requiring us to successfully recover the lower part of $\boldsymbol{\eta}$ first. We refer to Table 3 for the measured results for this attack stage, leading to successful recovery of the upper $n - k$ entries in $\boldsymbol{\eta}$ in all cases. Please note that the results here are significantly better than they were for the first attack stage allowing us to recover most of the upper $n - k$ values in a single attack round. In order to ensure that each value is actually recovered correctly, we utilize a second and a third round leading us to distinct results in our majority voting approach. Please note that one can easily use the remaining rounds from one signature generation to boost the results significance as they had to be measured for the first stage of the attack anyways. In the case of R-

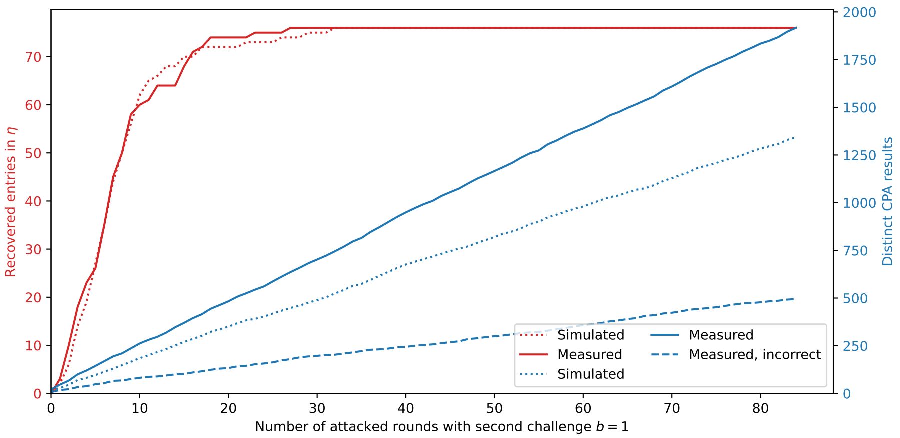

Fig. 6: Number of coefficients in $\boldsymbol{\eta}$ recovered per round of the ID-Protocol for R-SDP 1 Fast

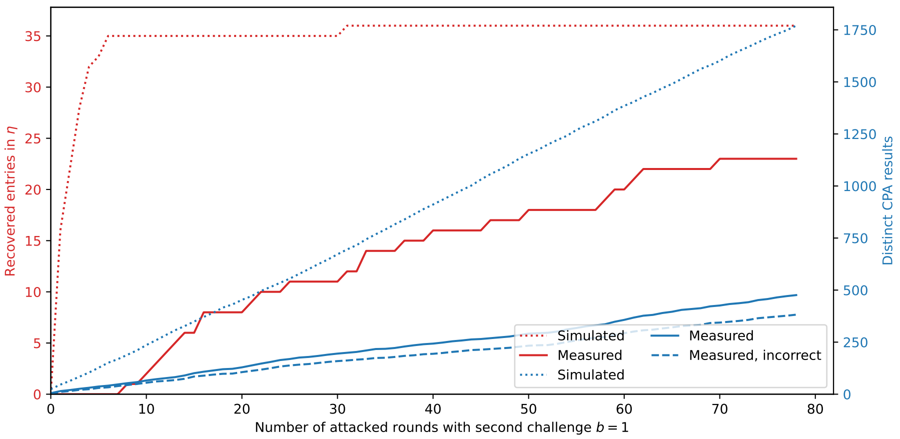

Fig. 7: Number of coefficients in $\boldsymbol{\eta}$ recovered per round of the ID-Protocol for R-SDP(G) 1 Fast

Table 2: Number of Rounds of the ID-Protocol necessary to recover the lower k entries in $\boldsymbol{\eta}$ as well as number of distinct CPA results and correctness rates of these results

Algorithm and Security Category	**Distinct CPA results/round**	**Frac. of correct CPA results**	**# rounds for lower k rec.**
CROSS-R-SDP **1**	23	74%	27
CROSS-R-SDP **3**	31	81%	59
CROSS-R-SDP **5**	42	83%	76
CROSS-R-SDP(G) **1**	27	13%	158
CROSS-R-SDP(G) **3**	32	29%	123
CROSS-R-SDP(G) **5**	31	43%	157

Table 3: Number of Rounds of the ID-Protocol necessary to recover the upper $n-k$ entries in $\boldsymbol{\eta}$ as well as number of distinct CPA results and correctness rates of these results

Algorithm and Security Category	**Distinct CPA results/round**	**Frac. of correct CPA results**	**# rounds for upper $n-k$ rec.**
CROSS-R-SDP **1**	50	99%	3
CROSS-R-SDP **3**	74	99.8%	3
CROSS-R-SDP **5**	99	99.9%	3
CROSS-R-SDP(G) **1**	9.8	79%	3
CROSS-R-SDP(G) **3**	18	80%	3
CROSS-R-SDP(G) **5**	25	82%	3

SDP(G), our target value $\boldsymbol{\zeta}$ is equal to the upper m entries in $\boldsymbol{\eta}$, meaning that a full recovery of $\boldsymbol{\eta}$ is also a full recovery of $\boldsymbol{\zeta}$.

4.4 Result Summary

As discussed above, we can recover all entries in $\boldsymbol{\eta}$ in the case of R-SDP using a limited number of rounds of the identification protocol from a single signature generation. In the case of R-SDP(G), we need at maximum two signature generations for RSDP(G) 1 Fast and also only a single signature generation for all other parameter sets to recover all entries in $\boldsymbol{\zeta}$.

5 Conclusion

In this work we present the first horizontal side-channel attack on CROSS. The attack is built upon the well established concept of horizontal attacks and exploits leakage from the syndrome computation inside CROSS' identification protocol which is executed several times in one signature generation allowing us to mount a separate attack on each computation. Only a single trace of a signature generation is required for a successful attack on all except one of the parameter sets of CROSS while we require at maximum two traces for a successful attack on the R-SDP(G) 1 Fast instance. The next natural step for this work is to consider different countermeasures usable to prevent this approach which for example include shuffling of the execution order of operations during a syndrome computation as well as masking of the syndrome computation's inputs. Another approach interesting for future work might be to improve the attack further as we currently discard a significant amount of samples because we rather focus on keeping the number of incorrect recoveries low but also leave the information of all discarded attack results unused.

Acknowledgments. The authors acknowledge the finanical support by the Federal Ministry of Education and Research of Germany in the project "PoQ-KIKI", project identification number 16KIS2065.

References

1. Stateless hash-based digital signature standard. National Institute of Standards and Technology, NIST FIPS PUB 205, U.S. Department of Commerce (Aug 2024). https://doi.org/10.6028/nist.fips.205
2. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D.: Status report on the third round of the nist post-quantum cryptography standardization process. Tech. rep. (Jul 2022). https://doi.org/10.6028/nist.ir.8413
3. Alagic, G., Bros, M., Ciadoux, P., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Liu, Y.K., Miller, C., et al.: Status report on the first round of the additional digital signature schemes for the nist post-quantum cryptography standardization process. Tech. rep. (2024). https://doi.org/10.6028/nist.ir.8528
4. Baldi, M., Barenghi, A., Bitzer, S., Karl, P., Manganiello, F., Pavoni, A., Pelosi, G., Santini, P., Schupp, J., Slaughter, F., Wachter-Zeh, A., Weger, V.: CROSS Documentation. [Online]. Available from: https://web.archive.org/web/20250122122245/http://www.cross-crypto.com/CROSS_Specification_v1.2.pdf, (Archived on 22 Jan. 2025) (2024), https://www.cross-crypto.com/CROSS_Specification_v1.2.pdf
5. Bauer, A., Jaulmes, E., Prouff, E., Reinhard, J.R., Wild, J.: Horizontal collision correlation attack on elliptic curves: – extended version –. Cryptography and Communications **7**(1), 91–119 (Oct 2014). https://doi.org/10.1007/s12095-014-0111-8
6. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004. pp. 16–29. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)

7. Chen, L., Jordan, S., Liu, Y.K., Moody, D., Peralta, R., Perlner, R., Smith-Tone, D.: Report on post-quantum cryptography. Tech. rep. (Apr 2016). https://doi.org/10.6028/nist.ir.8105
8. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) Information and Communications Security. pp. 46–61. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
9. Kannwischer, M.J., Pessl, P., Primas, R.: Single-trace attacks on keccak. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(3), 243–268 (Jun 2020). https://doi.org/10.13154/tches.v2020.i3.243-268, https://tches.iacr.org/index.php/TCHES/article/view/8590
10. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134 (Nov 1994). https://doi.org/10.1109/SFCS.1994.365700
11. Walter, C.D.: Sliding windows succumbs to big mac attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems — CHES 2001. pp. 286–299. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)