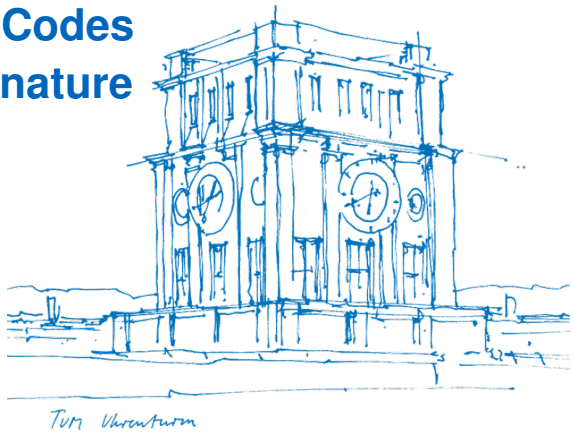
Chair of Security in Information Technology
TUM School of Computation, Information and Technology
Technical University of Munich

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

Jonas Schupp, Georg Sigl

Technical University of Munich
TUM School of Computation, Information and Technology

April 2, 2025



Outline

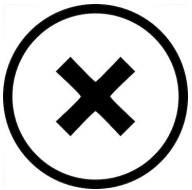
- Introduction and High-Level Overview on CROSS
- Recovering the Long-Term Secret from the Ephemeral Secret
- A Side-Channel Attack on the Syndrome Computation
- Applying the Attack to a Full Signing Operation
- Summary and Outlook

Outline

- Introduction and High-Level Overview on CROSS
- Recovering the Long-Term Secret from the Ephemeral Secret
- A Side-Channel Attack on the Syndrome Computation
- Applying the Attack to a Full Signing Operation
- Summary and Outlook

CROSS: Codes & Restricted Objects Signature Scheme [1]

- **Based on variants of the Syndrome Decoding Problem**
  - ✓ Compact representations
  - ✓ Efficient arithmetic
- **Fiat-Shamir Transform on Zero-Knowledge Protocol**
  - ✓ Trade-off potential: signature size & performance
- **Simple and efficient Operations**
  - ✓ Rather low implementation complexity
  - ✓ Possible to squeeze into microcontrollers



https://www.cross-crypto.com/

Attack Properties

- **Horizontal Attack [2–4] on Syndrome Computation**
  - ✓ Horizontal CPA [5] on Matrix-Vector Multiplication
  - ✓ Restricted hypothesis space due to underlying problem

- **Targets single round of the underlying ID-Protocol**
  - ✓ One attack per round of the ID-Protocol
  - ✓ Multiple useable attack targets per signature

- **Full recovery of a representation of the secret key (η)**
  - ✓ Only one signature generation necessary for all parameter sets and security levels except:
  - ✓ RSDP(G)-1 Fast requires two signing operations

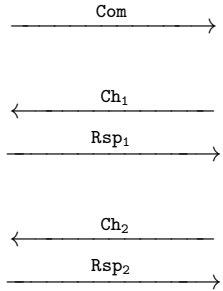
Generic 5-pass Scheme

PROVER

VERIFIER

Holds the secret key

Holds the public key



$\texttt{Ch}_1 \xleftarrow{\$} C_1$

$\texttt{Ch}_2 \xleftarrow{\$} C_2$

Return $\texttt{Out} \in \{0; 1\}$

CROSS – Identification Scheme

Private Key: restricted vector $\boldsymbol{e} \in G$
Public Key: group $G \leq \mathbb{E}^n$, parity-check matrix $\boldsymbol{H}$, syndrome $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^\top$

PROVER		VERIFIER
Sample $\texttt{Seed} \xleftarrow{\$} \{0;1\}^\lambda$, $(\boldsymbol{e}', \boldsymbol{u}') \xleftarrow{\texttt{Seed}} G \times \mathbb{F}_p^n$		
Compute $\boldsymbol{d} \in G$ such that $\boldsymbol{d} \star \boldsymbol{e}' = \boldsymbol{e}$ \\$\boldsymbol{d}$ is uniformly random over G		
Set $\boldsymbol{u} = \boldsymbol{d} \star \boldsymbol{u}'$ and $\widetilde{\boldsymbol{s}} = \boldsymbol{u}\boldsymbol{H}^\top$		
Set $c_0 = \mathsf{Hash}(\widetilde{\boldsymbol{s}}, \boldsymbol{d})$, $c_1 = \mathsf{Hash}(\boldsymbol{u}', \boldsymbol{e}')$ \\Commitments		
	$\xrightarrow{(c_0, c_1)}$	
	$\xleftarrow{\beta}$	Sample $\beta \xleftarrow{\$} \mathbb{F}_p^*$
Compute $\boldsymbol{y} = \boldsymbol{u}' + \beta \boldsymbol{e}'$ \\Uniformly random over $\mathbb{F}_p$		
Set $h = \mathsf{Hash}(\boldsymbol{y})$ \\First response		
	$\xrightarrow{h}$	
		Sample $b \xleftarrow{\$} \{0, 1\}$
	$\xleftarrow{b}$	
If $b = 0$, set $\texttt{rsp} = (\boldsymbol{y}, \boldsymbol{d})$ \\Second response (the larger one)		
If $b = 1$, set $\texttt{rsp} = \texttt{Seed}$ \\Second response (the shorter one)		
	$\xrightarrow{\texttt{rsp}}$	
		Verify c_b using $\texttt{rsp}$

Outline

- ⊗ Introduction and High-Level Overview on CROSS
- ⊗ Recovering the Long-Term Secret from the Ephemeral Secret
- ⊗ A Side-Channel Attack on the Syndrome Computation
- ⊗ Applying the Attack to a Full Signing Operation
- ⊗ Summary and Outlook

Recovering a representation of the secret key

Procedural Description

Sample $\mathsf{Seed} \xleftarrow{\$} \{0;1\}^{\lambda}, \quad (\boldsymbol{e}', \boldsymbol{u}') \xleftarrow{\mathsf{Seed}} G \times \mathbb{F}_p^n$
Compute $\boldsymbol{d} \in G$ such that $\boldsymbol{d} \star \boldsymbol{e}' = \boldsymbol{e}$
Set $\boldsymbol{u} = \boldsymbol{d} \star \boldsymbol{u}'$ and $\widetilde{\boldsymbol{s}} = \boldsymbol{u}\boldsymbol{H}^{\top}$

Recovering a representation of the secret key

Procedural Description

Sample $\texttt{Seed} \xleftarrow{\$} \{0;1\}^{\lambda}$, $(\boldsymbol{e}', \boldsymbol{u}') \xleftarrow{\texttt{Seed}} G \times \mathbb{F}_p^n$
Compute $\boldsymbol{d} \in G$ such that $\boldsymbol{d} \star \boldsymbol{e}' = \boldsymbol{e}$
Set $\boldsymbol{u} = \boldsymbol{d} \star \boldsymbol{u}'$ and $\widetilde{\boldsymbol{s}} = \boldsymbol{u}\boldsymbol{H}^\top$

Pseudocode

$\boldsymbol{\eta}_i', \boldsymbol{u}_i' \leftarrow \mathsf{CSPRNG}(\texttt{Seed}[i]||\texttt{Salt}||i+c,)$
$\boldsymbol{\sigma}_i \leftarrow \boldsymbol{\eta} - \boldsymbol{\eta}_i'$
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
$\tilde{\mathbf{s}} \leftarrow \boldsymbol{u}\mathbf{H}^\top$

Recovering a representation of the secret key

Procedural Description

Sample $\mathtt{Seed} \xleftarrow{\$} \{0;1\}^\lambda, \quad (\boldsymbol{e}', \boldsymbol{u}') \xleftarrow{\mathtt{Seed}} G \times \mathbb{F}_p^n$
Compute $\boldsymbol{d} \in G$ such that $\boldsymbol{d} \star \boldsymbol{e}' = \boldsymbol{e}$
Set $\boldsymbol{u} = \boldsymbol{d} \star \boldsymbol{u}'$ and $\widetilde{\boldsymbol{s}} = \boldsymbol{u}\boldsymbol{H}^\top$

Pseudocode

$\boldsymbol{\eta}_i', \boldsymbol{u}_i' \leftarrow \mathsf{CSPRNG}\,(\mathtt{Seed}[i]||\mathtt{Salt}||i+c,)$
$\boldsymbol{\sigma}_i \leftarrow \boldsymbol{\eta} - \boldsymbol{\eta}_i'$
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
$\tilde{\mathbf{s}} \leftarrow \boldsymbol{u}\mathbf{H}^\top$

Recovery

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

Outline

- Introduction and High-Level Overview on CROSS
- Recovering the Long-Term Secret from the Ephemeral Secret
- A Side-Channel Attack on the Syndrome Computation
- Applying the Attack to a Full Signing Operation
- Summary and Outlook

Implementation of the Syndrome Computation

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

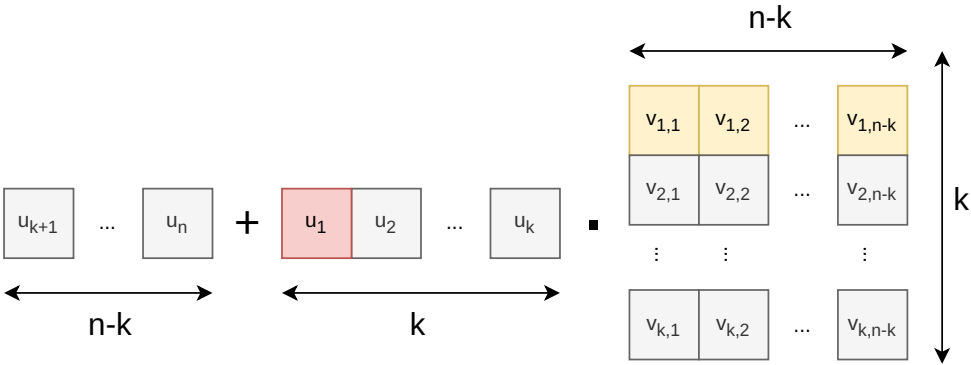
$$\tilde{\mathbf{s}} = \mathbf{u}\left[\mathbf{V}_{tr}\mathbf{I}_{n-k}\right]^\top$$

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

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

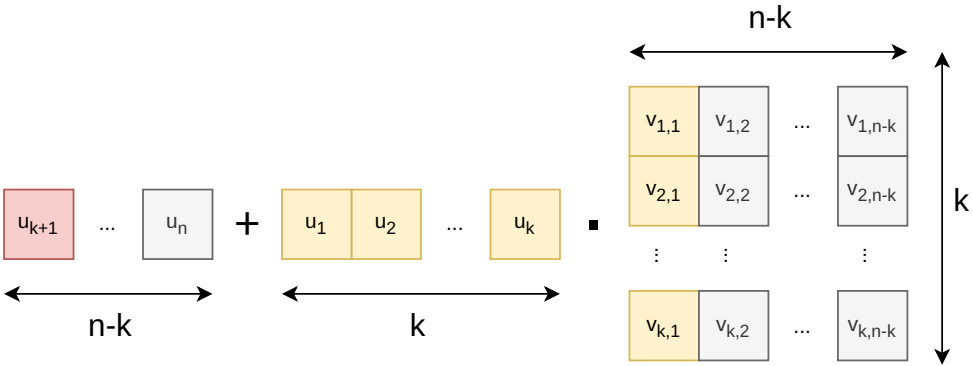
A CPA Style Attack on the Syndrome Computation (1)

Attack on the lower k entries of u



A CPA Style Attack on the Syndrome Computation (2)

Attack on the upper $n - k$ entries of u



Restricting the hypothesis space

RSDP

$$p = 127$$
$$z = 7$$

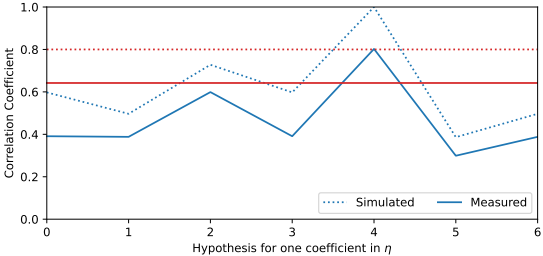
RSDP(G)

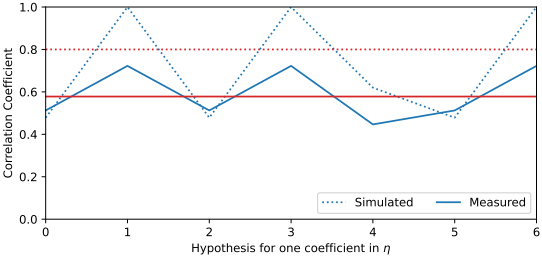
$$p = 509$$
$$z = 127$$

$$\mathbf{u} \in \mathbb{F}_p^n$$
$$\mathbf{V}_{tr} \in \mathbb{F}_p^{(n-k)\times k}$$
$$\boldsymbol{v} \in \mathbb{F}_p^n$$
$$\boldsymbol{\sigma} \in \mathbb{F}_z^n$$

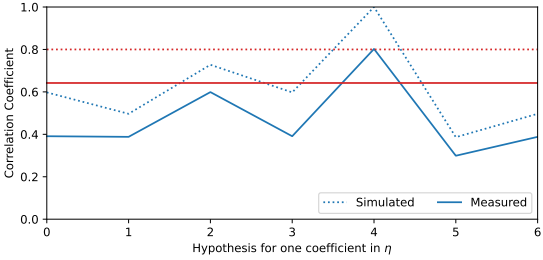
but we know $\mathbf{u}'$ from the signature, $\boldsymbol{v} = g^{\boldsymbol{\sigma}}$ and $\boldsymbol{u} = \boldsymbol{v} \star \boldsymbol{u}'$.
$\Rightarrow$ only z hypotheses for each entry in u.

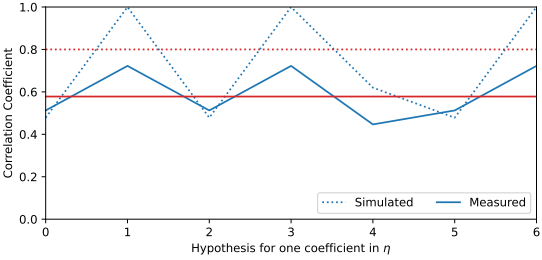
CPA results for single coefficients

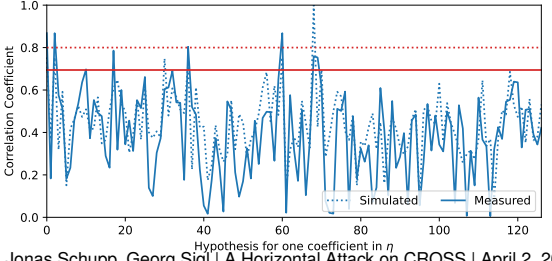


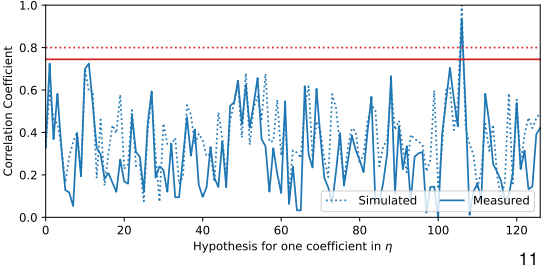


CPA results for single coefficients









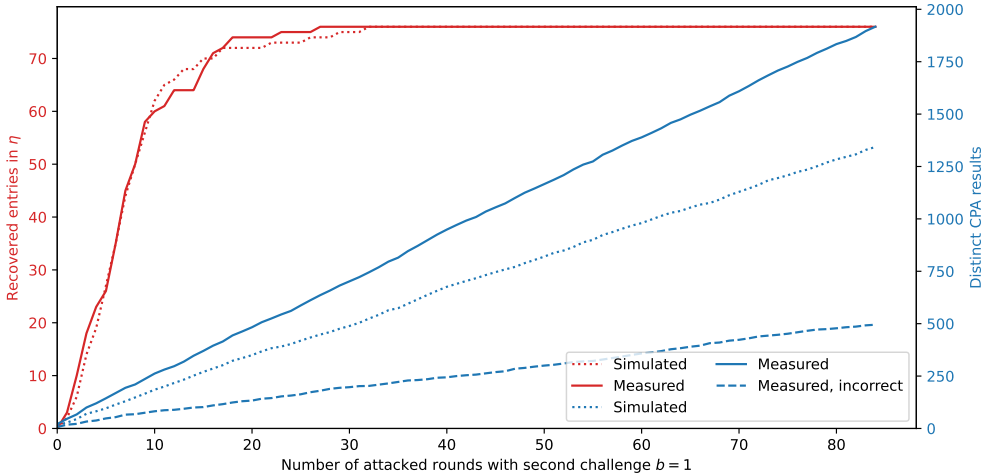
Outline

- ⊗ Introduction and High-Level Overview on CROSS
- ⊗ Recovering the Long-Term Secret from the Ephemeral Secret
- ⊗ A Side-Channel Attack on the Syndrome Computation
- ⊗ Applying the Attack to a Full Signing Operation
- ⊗ Summary and Outlook

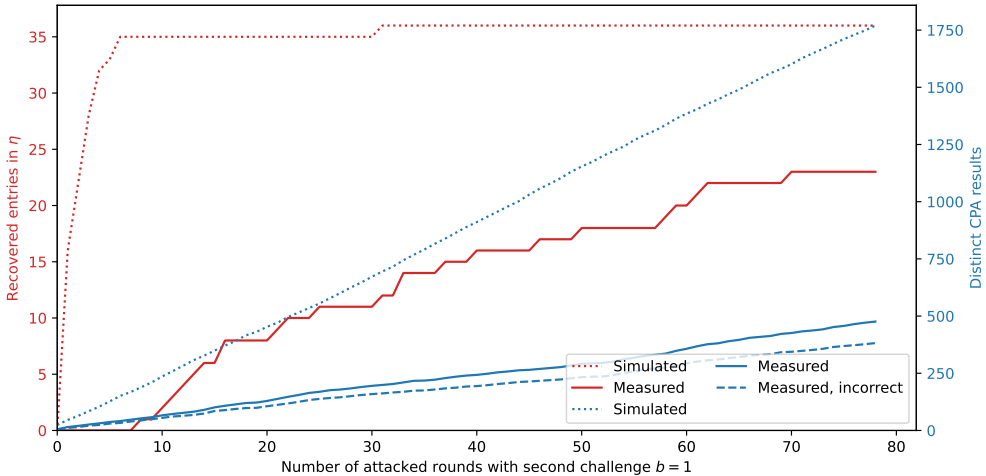
Applying the attack to a full signing operation (1)

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

Applying the attack to a full signing operation (2)



Applying the attack to a full signing operation (3)



Overall 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

Overall results (2)

Algorithm and	Distinct CPA	Frac. of correct	# rounds for
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

Outline

- ⊗ Introduction and High-Level Overview on CROSS
- ⊗ Recovering the Long-Term Secret from the Ephemeral Secret
- ⊗ A Side-Channel Attack on the Syndrome Computation
- ⊗ Applying the Attack to a Full Signing Operation
- ⊗ Summary and Outlook

Summary and Outlook

- Unsupervised CPA-Style Attack on CROSS
  - ✓ Attack exploits leakage from syndrome computation
  - ✓ Full key recovery from a single signing operation for all except one parameter sets

Summary and Outlook

- Unsupervised CPA-Style Attack on CROSS
  - ✓ Attack exploits leakage from syndrome computation
  - ✓ Full key recovery from a single signing operation for all except one parameter sets

- Outlook
  - ✗ Countermeasures?
  - ✗ Stronger (e.g. template based) attacks?

Summary and Outlook

- Unsupervised CPA-Style Attack on CROSS
  - ✓ Attack exploits leakage from syndrome computation
  - ✓ Full key recovery from a single signing operation for all except one parameter sets

- Outlook
  - ✗ Countermeasures?
  - ✗ Stronger (e.g. template based) attacks?

Thank you for your attention!

References

[1] M. Baldi et al. *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.

[2] C. Clavier et al. “Horizontal Correlation Analysis on Exponentiation”. In: *Information and Communications Security*. Ed. by M. Soriano, S. Qing, and J. López. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 46–61.

[3] A. Bauer et al. “Horizontal collision correlation attack on elliptic curves: – Extended Version –”. In: *Cryptography and Communications* 7.1 (Oct. 2014), pp. 91–119.

[4] C. D. Walter. “Sliding Windows Succumbs to Big Mac Attack”. In: *Cryptographic Hardware and Embedded Systems — CHES 2001*. Ed. by Ç. K. Koç, D. Naccache, and C. Paar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 286–299.

[5] E. Brier, C. Clavier, and F. Olivier. “Correlation Power Analysis with a Leakage Model”. In: *Cryptographic Hardware and Embedded Systems - CHES 2004*. Ed. by M. Joye and J.-J. Quisquater. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 16–29.

Backup: CROSS Sign Pesudocode

Algorithm 1: Sign(pri, Msg), taken from [4]

Data: λ: 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
$\mathcal{B}'_w$: 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
dsc: a fixed constant, greater than t employed to obtain domain separation

Input: pri: private key constituted of $\mathtt{Seed}_{\mathtt{sk}} \in \{0,1\}^{2\lambda}$
Msg: message to be signed $\mathtt{Msg} \in \{0,1\}^*$

Output: Signature Signature

Begin

// Key material expansion

$\boldsymbol{\eta}, \boldsymbol{\zeta}, \mathbf{H}, \mathbf{M}_G \leftarrow \text{ExpandPrivateSeed}(\mathtt{Seed}_{\mathtt{sk}})$

$\boldsymbol{\eta}, \mathbf{H} \leftarrow \text{ExpandPrivateSeed}(\mathtt{Seed}_{\mathtt{sk}})$

// Computation of commitments

$\mathtt{Mseed} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathtt{Salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$

$(\mathtt{Seed}[0], \ldots, \mathtt{Seed}[t-1]) \leftarrow \text{SeedTreeLeaves}(\mathtt{Mseed}, \mathtt{Salt})$

for $i \leftarrow 0$ **to** $t-1$ **do**

$\boldsymbol{\zeta}', \boldsymbol{u}_i \leftarrow \text{CSPRNG}\,(\mathtt{Seed}[i]||\mathtt{Salt}||i+c,)$

$\boldsymbol{\delta}_i \leftarrow \boldsymbol{\zeta} - \boldsymbol{\zeta}'$

$\boldsymbol{\eta}'_i \leftarrow \boldsymbol{\zeta}' \mathbf{M}_G$

$\boldsymbol{\eta}'_i, \boldsymbol{u}'_i \leftarrow \text{CSPRNG}\,(\mathtt{Seed}[i]||\mathtt{Salt}||i+c,)$

$\boldsymbol{\sigma}_i \leftarrow \boldsymbol{\eta} - \boldsymbol{\eta}'_i$

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

$\bar{\mathbf{s}} \leftarrow \boldsymbol{u}\mathbf{H}^\top$

$\mathtt{cmt}_0[i] \leftarrow \text{Hash}(\bar{\mathbf{s}}||\boldsymbol{\delta}_i||\mathtt{Salt}||i+c+\mathtt{dsc})$

$\mathtt{cmt}_0[i] \leftarrow \text{Hash}(\bar{\mathbf{s}}||\boldsymbol{\sigma}_i||\mathtt{Salt}||i+c+\mathtt{dsc})$

$\mathtt{cmt}_1[i] \leftarrow \text{Hash}(\mathtt{Seed}[i]||\mathtt{Salt}||i+c+\mathtt{dsc})$

end

$\mathtt{d}_0 \leftarrow \text{MerkleRoot}(\mathtt{cmt}_0[0], \ldots, \mathtt{cmt}_0[t-1])$

$\mathtt{d}_1 \leftarrow \text{Hash}(\mathtt{cmt}_1[0] \;||\; \ldots \;||\; \mathtt{cmt}_1[t-1])$

$\mathtt{d}_{01} \leftarrow \text{Hash}(\mathtt{d}_0 \;||\; \mathtt{d}_1)$

// First challenge vector extraction

$\mathtt{d}_m \leftarrow \text{Hash}(\mathtt{Msg})$

$\mathtt{d}_\beta \leftarrow \text{Hash}(\mathtt{d}_m||\mathtt{d}_{01}||\mathtt{Salt})$

$\mathtt{beta} \leftarrow \text{CSPRNG}\,(\mathtt{d}_\beta, (\mathbb{F}_p^*)^t)$

// 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

$\mathbf{y}_i \leftarrow \boldsymbol{u}'_i + \mathtt{beta}[i]\mathbf{e}'_i$

end

// Second challenge vector extraction

$\mathtt{d}_b \leftarrow \text{Hash}(\mathbf{y}_0||\ldots||\mathbf{y}_{t-1}||\mathtt{d}_\beta)$

$\mathbf{b} \leftarrow \text{CSPRNG}\,(\mathtt{d}_b, \mathcal{B}'_{(w)})$

// Computation of second round of responses

$\mathtt{MerkleProofs} \leftarrow \text{MerkleProof}(\{\mathtt{cmt}_0[0], \ldots, \mathtt{cmt}_0[t-1]\}, \mathbf{b})$

$\mathtt{SeedPath} \leftarrow \text{SeedTreePaths}(\mathtt{Mseed}, \mathbf{b})$

// Signature composition

$\mathtt{rsp}_0 \leftarrow (\mathbb{F}_p^n \times \mathbb{F}_z^m)^{t-w}$; $\mathtt{rsp}_0 \leftarrow (\mathbb{F}_p^n \times \mathbb{F}_z^n)^{t-w}$

$\mathtt{rsp}_1 \leftarrow (\{0,1\}^{2\lambda})^{t-w}$ // empty array

$j \leftarrow 0$

for $i \leftarrow 0$ **to** $t-1$ **do**

if $(\mathbf{b}[i] = 0)$ **then**

// $\mathtt{cmt}_0[i]$ is recomputed by the verifier, $\mathtt{cmt}_1[i]$ must be sent

$\mathtt{rsp}_0[j] \leftarrow (\mathbf{y}_i, \boldsymbol{\delta}_i)$; $\mathtt{rsp}_0[j] \leftarrow (\mathbf{y}_i, \boldsymbol{\sigma}_i)$

$\mathtt{rsp}_1[j] \leftarrow \mathtt{cmt}_1[i]$

$j \leftarrow j+1$

end

end

$\mathtt{Signature} \leftarrow \mathtt{Salt} \;||\; \mathtt{d}_{01} \;||\; \mathtt{d}_b \;||\; \mathtt{MerkleProofs} \;||\; \mathtt{SeedPath} \;||\; \mathtt{rsp}_0 \;||\; \mathtt{rsp}_1$

// all Signature components are encoded as binary strings

return Signature

end