RSA Conference2020

San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

SESSION ID: **CRYP-T10**

Modeling Memory Faults in Signature and Authenticated Encryption Schemes

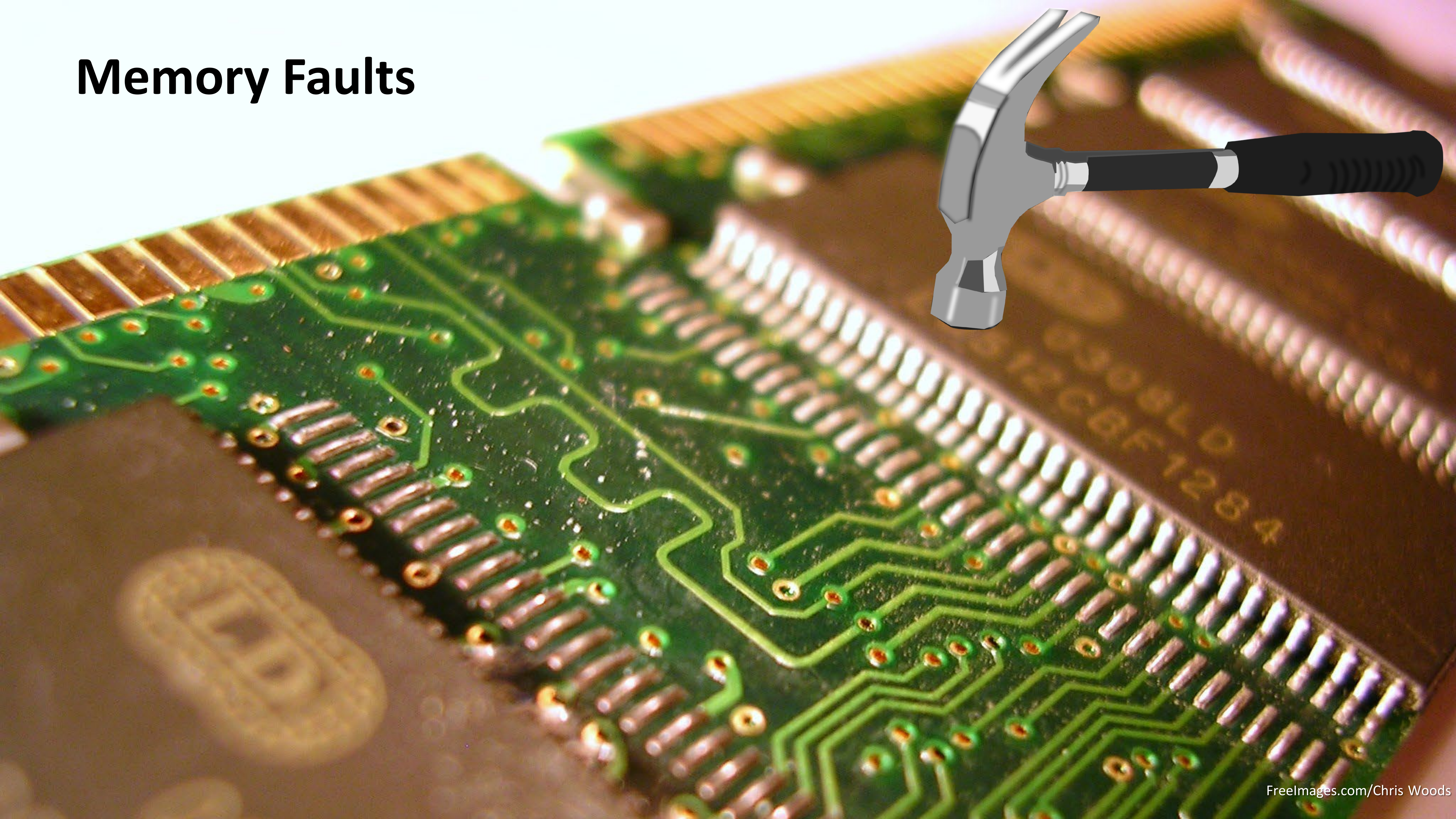
Felix Günther

Postdoctoral Researcher
Department of Computer Science, ETH Zurich, Switzerland

joint work with **Marc Fischlin (TU Darmstadt)**

ETH zürich UC San Diego TECHNISCHE UNIVERSITÄT DARMSTADT DFG Deutsche Forschungsgemeinschaft German Research Foundation

Memory Faults



FreeImages.com/Chris Woods

What About the Code?

```
rivate $host;
private $username;
private $password;
private $database;
private $charset;
static private $link = null;
static public function connect()
self::$link = mysql_connect(self::$host, self::
self::$link)
new mysqlException("Cannot connect to
```

FreeImages.com/Gabor Heja

The Cryptographic Perspective

Deterministic ECDSA

```
Sign_det-ECDSA(sk, m)
  r ← Hash(sk, m)
  R ← f(rG)  mod q
  s ← (H(m) + sk R)/r mod q
  return (R, s)
```

What about faults?

Signature security (EUF-CMA)

```
$\mathsf{Expt}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}(1^\lambda)$:
  $(sk, pk) \overset{\$}{\leftarrow} \mathsf{KGen}(1^\lambda)$
  $Q \leftarrow \emptyset$
  $(m^*, \sigma^*) \overset{\$}{\leftarrow} \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(1^\lambda, pk)$
  return 1 iff $(m^*, *) \notin Q$
      and $\mathsf{Verify}(pk, m^*, \sigma^*) = 1$

$\mathcal{O}_{\mathsf{Sign}}(m)$:
  $\sigma \overset{\$}{\leftarrow} \mathsf{Sign}(sk, m)$
  $Q \leftarrow Q \cup \{(m, \sigma)\}$
  return $\sigma$
```

Models Matter

- Deterministic ECDSA (& co.) succumb to rowhammer-style faults [PSSLR @ IEEE EuroS&P 2018]

$$(R_0, s_0): \quad H(m) + sk\ R_0 = Hash(sk,\ m)\, s_0$$

$$(\mathbf{R}^{!}, \mathbf{s}^{!}): \quad H(m) + sk\ \mathbf{R}^{!} = Hash(sk,\ m)\, \mathbf{s}^{!}$$

$$\mathbf{sk} = H(m) \ / \ ((R_0 - R^{!})\, s_0 \ / \ (s_0 - s^{!}) \ - \ R_0)$$

- We know for long that faults can have devastating effects on crypto operations at software level [BDL @ Eurocrypt 1997]
- But how to assess fault *resilience* in provable-security manner?

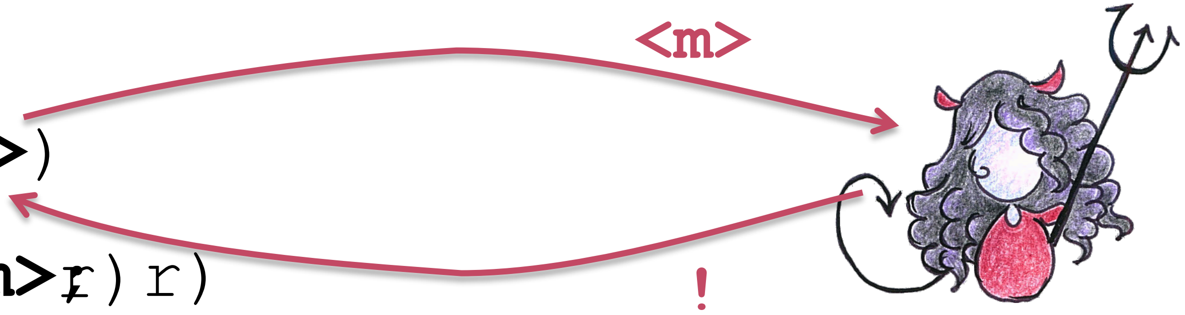
Prior Work

- Faults in circuits
  [IPSW06]
- Tailored provable-security models (e.g., for RSA)
  [CM09, BDFGTZ14, FGLTZ12]
- Related-key attack (RKA) security
  [BK04, GLMMR04]
- Hedged randomness in Fiat-Shamir-type signatures under faults
  [AOTZ19]

A Generic Framework for Fault Resilience in Security Models

Modeling Fault Resilience

```
$\underline{\mathrm{Sign}_{dr}(sk,\ \mathbf{m})}$
  r ← Hash(sk, <m>)
  s ← $\mathrm{Sign}_r$(sk, <m>, r)
  return s
```



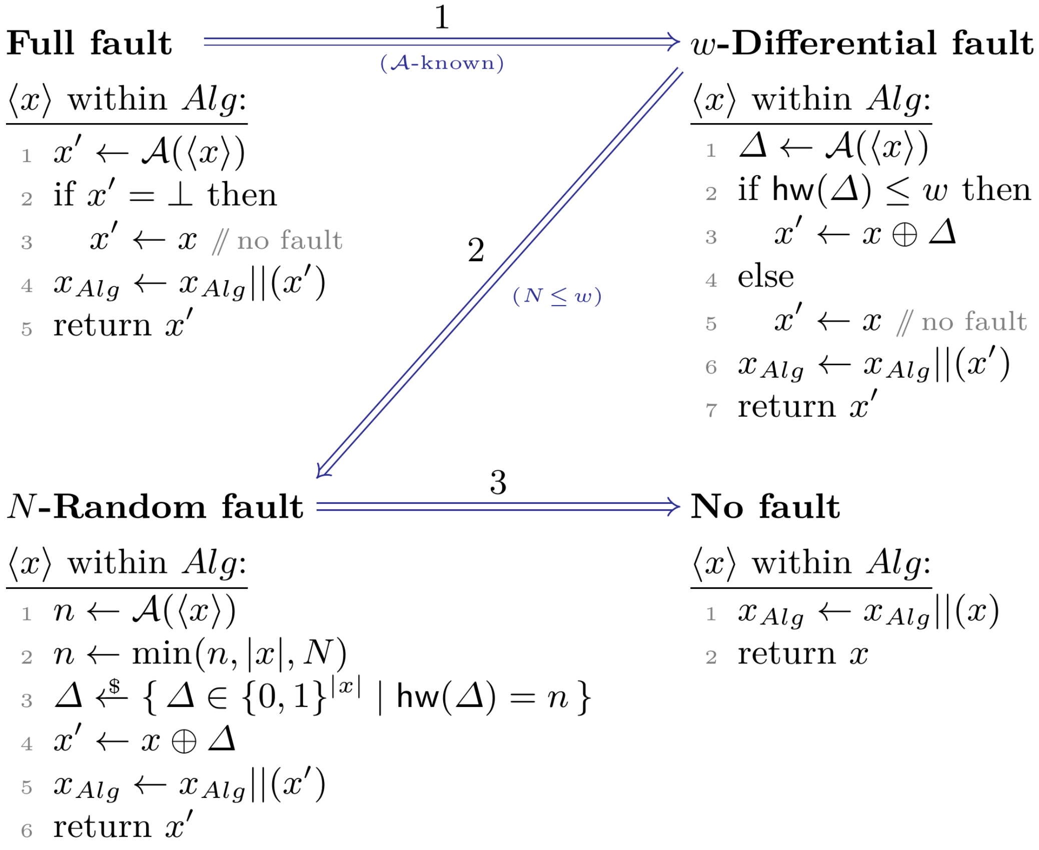
- **augmented code**, indicating faultable memory variables
- **callbacks** to adversary: may change values of variable readings

Generic Fault Types

Flexible callbacks

- Full faults
  adversary controls variable completely
- Differential faults
  adversary can flip *w* selected bits
- Random faults
  adversary can flip *N* random bits
- No fault
  (baseline)

Forming a hierarchy



Full fault

```
⟨$x$⟩ within $Alg$:
$x' \leftarrow \mathcal{A}(\langle x \rangle)$
if $x' = \perp$ then
    $x' \leftarrow x$ // no fault
$x_{Alg} \leftarrow x_{Alg} || (x')$
return $x'$
```

w-Differential fault

```
⟨$x$⟩ within $Alg$:
$\Delta \leftarrow \mathcal{A}(\langle x \rangle)$
if $\mathsf{hw}(\Delta) \leq w$ then
    $x' \leftarrow x \oplus \Delta$
else
    $x' \leftarrow x$ // no fault
$x_{Alg} \leftarrow x_{Alg} || (x')$
return $x'$
```

N-Random fault

```
⟨$x$⟩ within $Alg$:
$n \leftarrow \mathcal{A}(\langle x \rangle)$
$n \leftarrow \min(n, |x|, N)$
$\Delta \xleftarrow{\$} \{ \Delta \in \{0,1\}^{|x|} \mid \mathsf{hw}(\Delta) = n \}$
$x' \leftarrow x \oplus \Delta$
$x_{Alg} \leftarrow x_{Alg} || (x')$
return $x'$
```

No fault

```
⟨$x$⟩ within $Alg$:
$x_{Alg} \leftarrow x_{Alg} || (x)$
return $x$
```

Fault Resilience for Signatures

Augmenting Signature Security

frEUF-CMA: Fault-resilience unforgeability

```
$\text{Sign}_{\text{dr}}$(sk, m)
  r ← Hash(sk, <m>)
  s ← $\text{Sign}_{\text{r}}$(sk, <m>; r)
  return s
```

- Essential question:

  Which message did the signer sign?

  = Which (m,s) is trivially learned?

- Answer: the message **m** (among all appearing in Sign) verifying with **s**
- If there‘s two such **m** → confusion → adversary declared successful

De-Randomized Signatures Are Not Fault-Resilient

```
$\underline{\text{Sign}_{dr}(sk,\ \mathbf{m})}$
  $r \leftarrow \text{Hash}(sk,\ \mathbf{<m>})$
  $s \leftarrow \text{Sign}_r(sk,\ \mathbf{<m>};\ r)$
  return s
```

1. Query O_{Sign} on m
   - no faults
   - obtain signature s on m
2. Query O_{Sign} on m
   - first **<m>**: do nothing
   - second **<m>**: flip bit (to m‘)
   - obtain signature s on m‘
3. Create new forgery due to re-used randomness r for signatures on m and m‘

Combining Randomization & De-Randomization

```
$\underline{\mathrm{Sign}_C(\mathrm{sk},\ \mathbf{m})}$
  $\mathbf{r}' \leftarrow_\$ \{0,1\}^\lambda$
  $\mathbf{r} \leftarrow \mathrm{Hash}(\mathrm{sk},\ \langle\mathbf{m}\rangle,\ \langle\mathbf{r}'\rangle)$
  $\mathrm{s} \leftarrow \mathrm{Sign}_r(\mathrm{sk},\ \langle\mathbf{m}\rangle;\ \langle\mathbf{r}\rangle)$
  return s
```

Combining security (provably)

- **de-randomization** for regular EUF-CMA security under bad randomness
- **randomization** for fault-resilient EUF-CMA security under differential faults on m, r, r‘

Confirms XEdDSA design in Signal protocol

RSAConference2020

Fault Resilience for Authenticated Encryption

A Similar Setting

- good randomness isn‘t always available
- nonce-based authenticated encryption (AE) to avoid randomness
- nonce-misuse resistance hedging against repeated nonces

- but what about faults?

SIV Mode of Operation: Synthetic IV [RS06]

Nonce-misue resistance …

```
$\underline{\mathrm{Enc}_{SIV}((K_1,K_2), \mathbf{N}, \mathbf{A}, \mathbf{m})}$
  IV ← PRF($K_1$, <N>|<A>|<m>)
  c  ← Enc($K_2$, <m>; <IV>)
  return (IV, c)
```

… but vulnerable to faults

1. Query O_{Enc} on (N=00..0,A,m)
   - no faults, obtain c_1 = c or $
2. Query O_{Enc} on (N=10..0,A,m)
   - **<N>** callback: flip 1st bit
   - obtain c_2 = c or *different* $
3. Distinguish by checking if $c_1 = c_2$

SIV$: Combining Randomization & De-Randomization

```
$\underline{\mathtt{Enc}_{\mathtt{SIV\$}}((\mathtt{K}_1,\mathtt{K}_2),\ \mathbf{N},\ \mathbf{A},\ \mathbf{m})}$
  $\mathtt{r} \leftarrow_{\$} \{0,1\}^{\lambda}$
  $\mathbf{IV} \leftarrow \mathtt{PRF}(\mathtt{K}_1,\ \mathbf{<N>}|\mathbf{<A>}|\mathbf{<m>}|\mathbf{<r>})$
  $\mathtt{c} \leftarrow \mathtt{Enc}(\mathtt{K}_2,\ \mathbf{<r>}|\mathbf{<m>};\ \mathbf{<IV>})$
  return (IV, c)
```

Combining security (provably)

- **synthetic IV approach** for nonce-misuse res. AE security under bad randomness
- **augmented randomness** for fault-resilient nm-res. AE security under diff. faults on N, A, m, r, IV

Fault-resilient AE mode translating signature concepts

Summary

- Introduced **generic model** for understanding **fault resilience** in computational **security proofs**
- Signatures
  - confirm fault attacks on de-randomized signatures
  - provable security of **combined randomization + de-randomization**

XEdDSA

- Authenticated encryption
  - fault-attack treatment of SIV mode of operation
  - propose **combined SIV$ mode** achieving fault resilience

Applying the Generic Fault Resilience Model

- Select your favorite crypto primitive
  - fault resilience model is generic
- Revise security definitions towards fault-resilient variant
  - What has to be taken care of when faults might happen in schemes?
- Augment scheme with faulting profile
  - different memory variables / algorithms may be differently vulnerable
- Assess provable fault-resilient security of augmented scheme

Summary

- Introduced **generic model** for understanding **fault resilience** in computational **security proofs**
- Signatures
  - confirm fault attacks on de-randomized signatures
  - provable security of **combined randomization + de-randomization**
- Authenticated encryption
  - fault-attack treatment of SIV mode of operation
  - propose **combined SIV$ mode** achieving fault resilience

Thank you!

mail@**felixguenther.info**