Modeling Memory Faults in Signature and Authenticated Encryption Schemes

Felix Günther

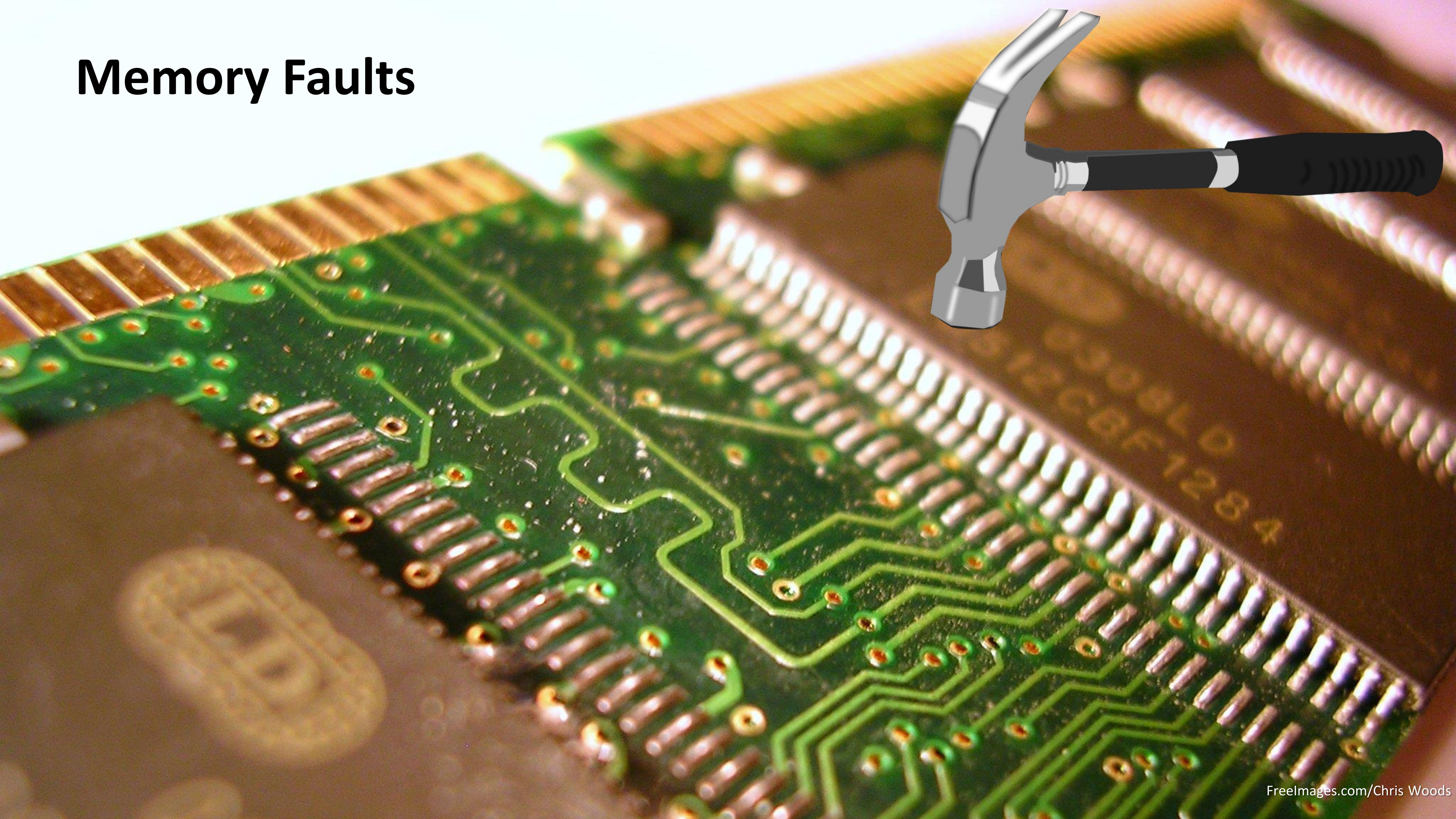
Postdoctoral Researcher
Department of Computer Science, ETH Zurich, Switzerland

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

RSA Conference 2020
San Francisco
February 25, 2020

Deutsche Forschungsgemeinschaft
German Research Foundation

Memory Faults



FreeImages.com/Chris Woods

What About the Code?

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)

$\mathsf{Expt}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}(1^\lambda)$:

$(sk, pk) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$

$Q \leftarrow \emptyset$

$(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(1^\lambda, pk)$

return 1 iff $(m^*, *) \notin Q$

and $\mathsf{Verify}(pk, m^*, \sigma^*) = 1$

$\mathcal{O}_{\mathsf{Sign}}(m)$:

$\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$

$Q \leftarrow Q \cup \{(m, \sigma)\}$

return σ

ETH zürich

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

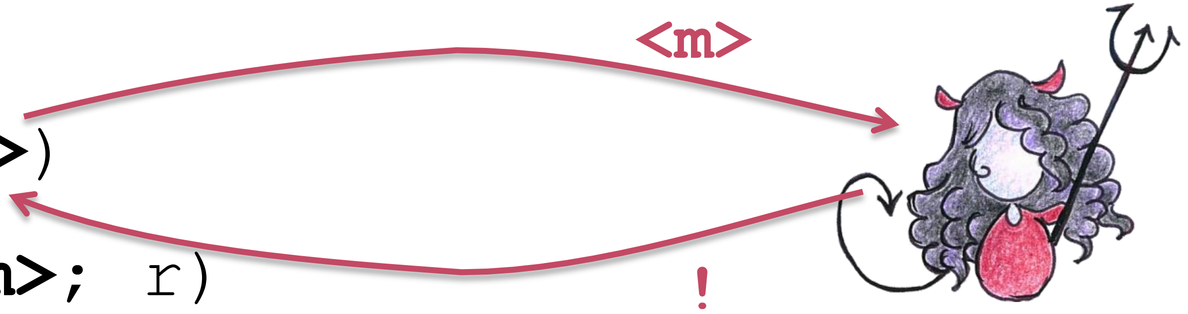
- Faults in circuits
  [IPSW06]
- Tailored provable-security models (e.g., for RSA)
  [CM09, BDFGTZ14, FGLTZ12]
- Related-key attack (RKA) security
  [BK04, GLMMR04]
- Hedged randomness in Fiat-Shamir-type signatures under faults
  [AOTZ19]

ETH zürich

A Generic Framework for
Fault Resilience in Security Models

Modeling Fault Resilience

```
Sign_dr(sk, m)
  r ← Hash(sk, <m>)
  s ← Sign_r(sk, <m>; r)
  return s
```



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

drawing by Giorgia Azzurra Marson

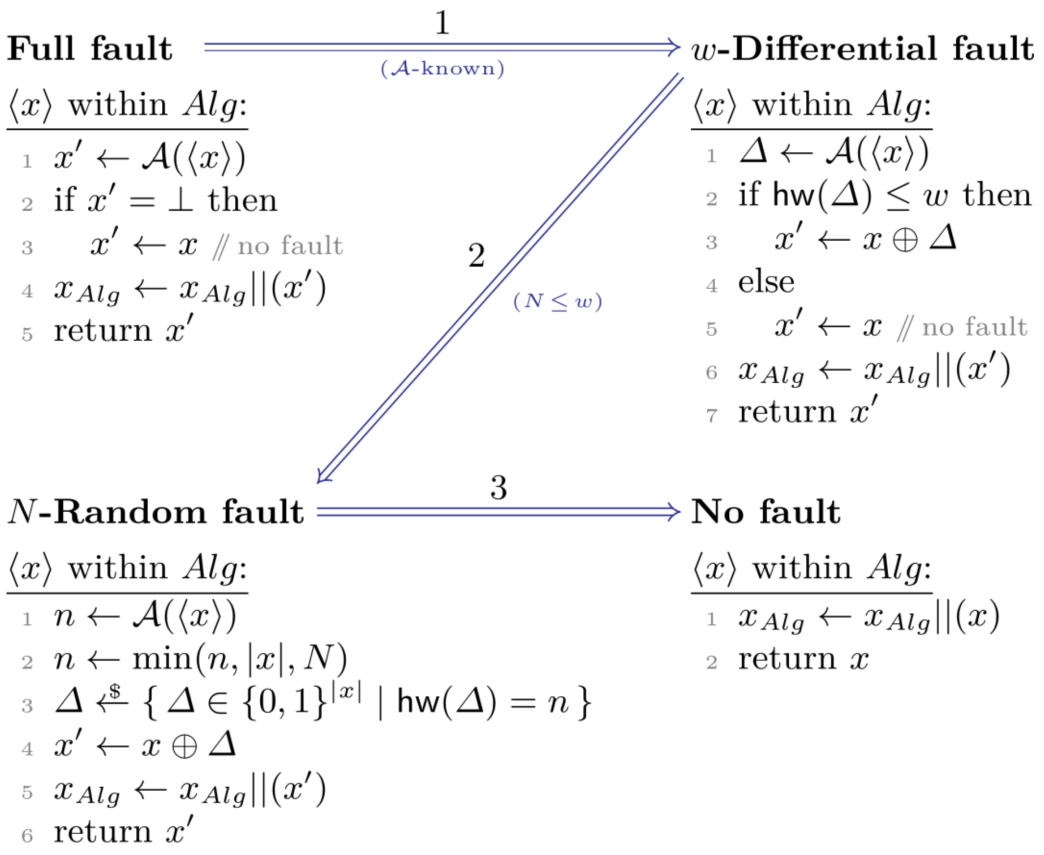
ETH zürich

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



Fault Resilience for Signatures

Augmenting Signature Security

frEUF-CMA: Fault-resilience unforgeability

```
Sign_dr(sk, m)
  r ← Hash(sk, <m>)
  s ← Sign_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

```
Sign_dr(sk, m)
  r ← Hash(sk, <m>)
  s ← Sign_r(sk, <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

```
Sign_c(sk, m)
  r' ←$ {0,1}^λ
  r ← Hash(sk, <m>, <r'>)
  s ← Sign_r(sk, <m>; <r>)
  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

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 …

```
Enc_SIV((K_1,K_2), N, A, 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

```
Enc_SIV$((K_1,K_2), N, A, m)
  r  ←$ {0,1}^λ
  IV ← PRF(K_1, <N>|<A>|<m>|<r>)
  c  ← Enc(K_2, <r>|<m>; <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!

ETH zürich