IF AND HOW IMPLEMENTATION ATTACKS SHAPE THE DESIGN OF LATTICE-BASED SIGNATURE SCHEMES

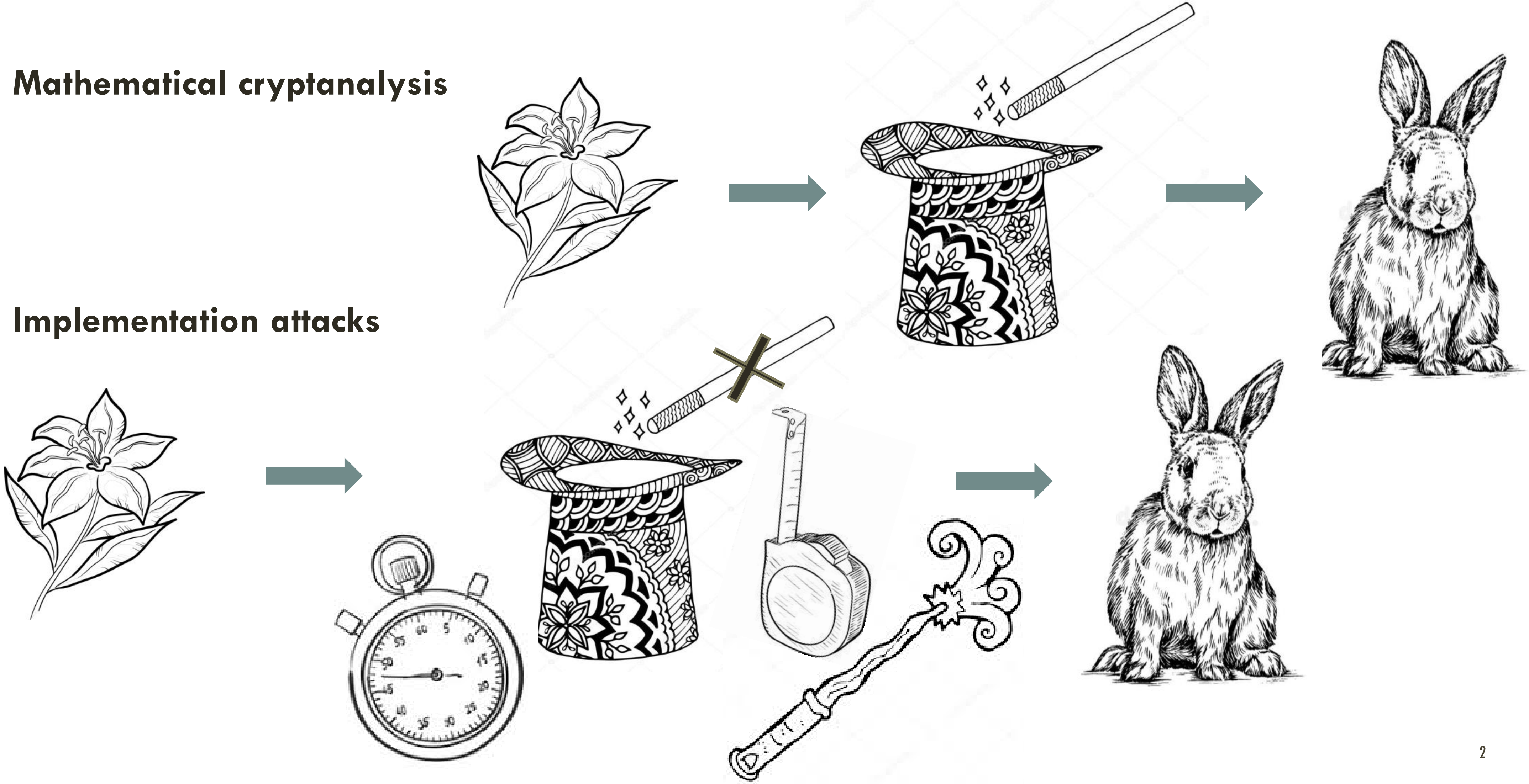
TECHNISCHE
UNIVERSITÄT
DARMSTADT

CROSSING

16th IMA International Conference
on Cryptography and Coding 2017
12/12/2017

Nina Bindel
TU Darmstadt

WHAT ARE IMPLEMENTATION ATTACKS?



Mathematical cryptanalysis

Implementation attacks

PASSIVE AND ACTIVE ATTACKS

Active Fault attacks	**Passive** Side-channel attacks
“allow to extract secret information by disturbing the cryptographic computation”	“monitor the behavior of the target device while executing”
Zeroing, skipping, Randomization faults	Timing, power, cache side channels

IMPLEMENTATION ATTACKS AGAINST LATTICE-BASED SIGNATURES IN THE LITERATURE

Year	Authors	IACR eprint	Type	Schemes
2012	Kamal and Youssef		FA	NTRUSign
2016	Espitau, Fouque, Gérard, and Tibouchi	2016/449	FA	GLP, BLISS, ring-TESLA, GPV-NTRU, PassSign
	Bindel, Buchmann, and Krämer	2016/415	FA	GLP, BLISS, ring-TESLA
	Groot Bruinderink, Hülsing, Lange, and Yarom	2016/300	Cache SC	BLISS
	Saarinen	2016/276	Cache SC	BLISS
	Pessl	2017/033	Cache SC	BLISS
2017	Bindel, Buchmann, Krämer, Mantel, Schickel, and Weber	2017/951	Cache SC	ring-TESLA
	Espitau, Fouque, Gerard, and Tibouchi	2017/505	(Power) SC	BLISS
	Pessl, Groot Bruinderink, and Yarom	2017/490	Cache SC	BLISS

Aren‘t implementation attacks only interesting for implementers?

Or are they also interesting for the designers of schemes?

OUTLINE

How fault attacks shape the design

Known attacks

Probabilistic
vs.
deterministic

Concrete examples: **qTESLA**
https://tesla.informatik.tu-darmstadt.de/de/tesla

How (cache-) side channels shape the design

Gaussian sampling

Analysis of cache side channels using program semantic

OUTLINE

How fault attacks shape the design

Known attacks

Probabilistic
vs.
deterministic

How (cache-) side channels shape the design

Gaussian sampling

Analysis of cache side channels using program semantic

RANDOMIZATION OF SMALL SECRET AND ERROR

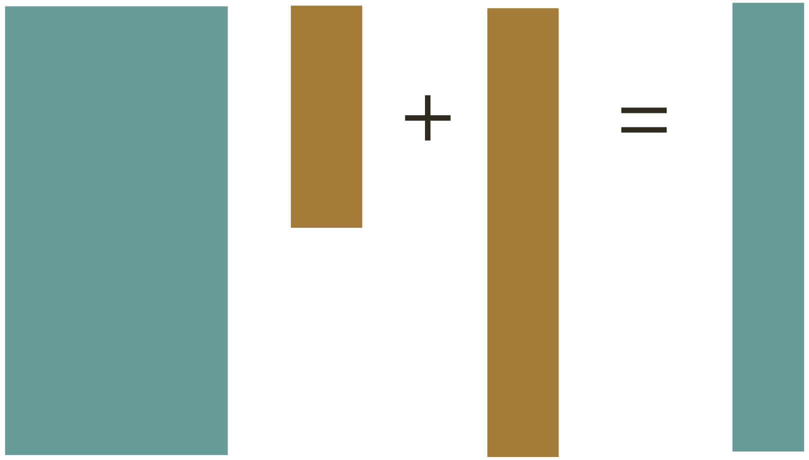
qTESLA
secret key: s,e $\leftarrow_\sigma \mathbb{Z}[x]/\langle x^n + 1\rangle$

public key: a $\leftarrow_\$ \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, b = a · s + e mod q

Possible alternative:
Binary LWE with s,e small coefficients

Problem: much easier to run randomization attack during signature generation [IACR eprint 2016/415]

LWE



A · s + e = b mod q

IDEA RANDOMIZATION ATTACK

- **1st Insert fault:** change one coeff. $s_i \in \{-1,0,1\}$ to $s_i' \in \{-1,0,1\}$
- **2nd Software computation:** find index i and determine value of s_i by “intelligent brute force”

Smaller interval of secret coeff.s

⟷

More efficient computation/attack

- ○ if s,e ←  → too many possibilitites for s_i → attack is not feasible
- ○ can also be prevented by implementing countermeasure

OUTLINE

How fault attacks shape the design

Known attacks

Probabilistic
vs.
deterministic

How (cache-) side channels shape the design

Gaussian sampling

Analysis of cache side channels using program semantic

DETERMINISTIC SIGNATURE QTESLA

m,
sk = (s, e, seed, a)

```
1. counter ← 0
2. rand ← PRF(seed, m)
3. y ← PRF(rand, counter)
4. $c \leftarrow H(\lfloor ay \rceil, m)$
5. $z \leftarrow y + sc$
6. if $ay - ec$ is not small enough:                (Correctness)
       counter++ and retry at step 1
7. if z is not small enough:                        (Security)
       counter++ and retry at step 1
8. return (z,c)
```

(z, c)

DETERMINISTIC VS PROBABILIST SIGNATURE

Advantages deterministic signature:

✓ Use different randomness for different messages

→ prevent attacks that exploit fixed randomness

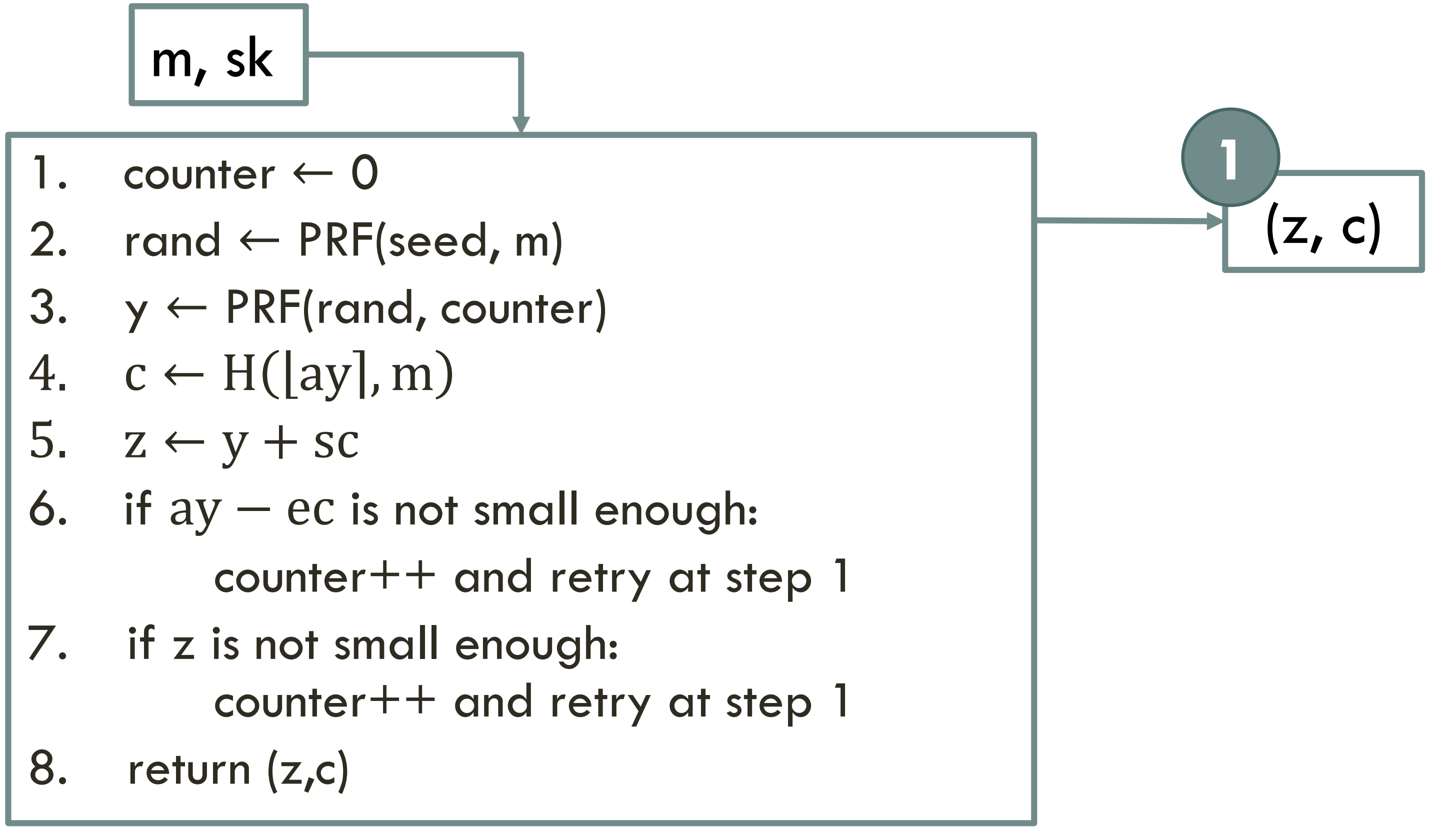
✓ No need of of high-quality randomness

→ easier to be implemented

BUT possible vulnerability to fault attack might be introduced….

FAULT ATTACK ON DETERMINISTC SIGNATURE

by Poddebniak, Somorovsky, Schinzel, Lochter, and Rösler [eprint 2017/1014]



m, sk

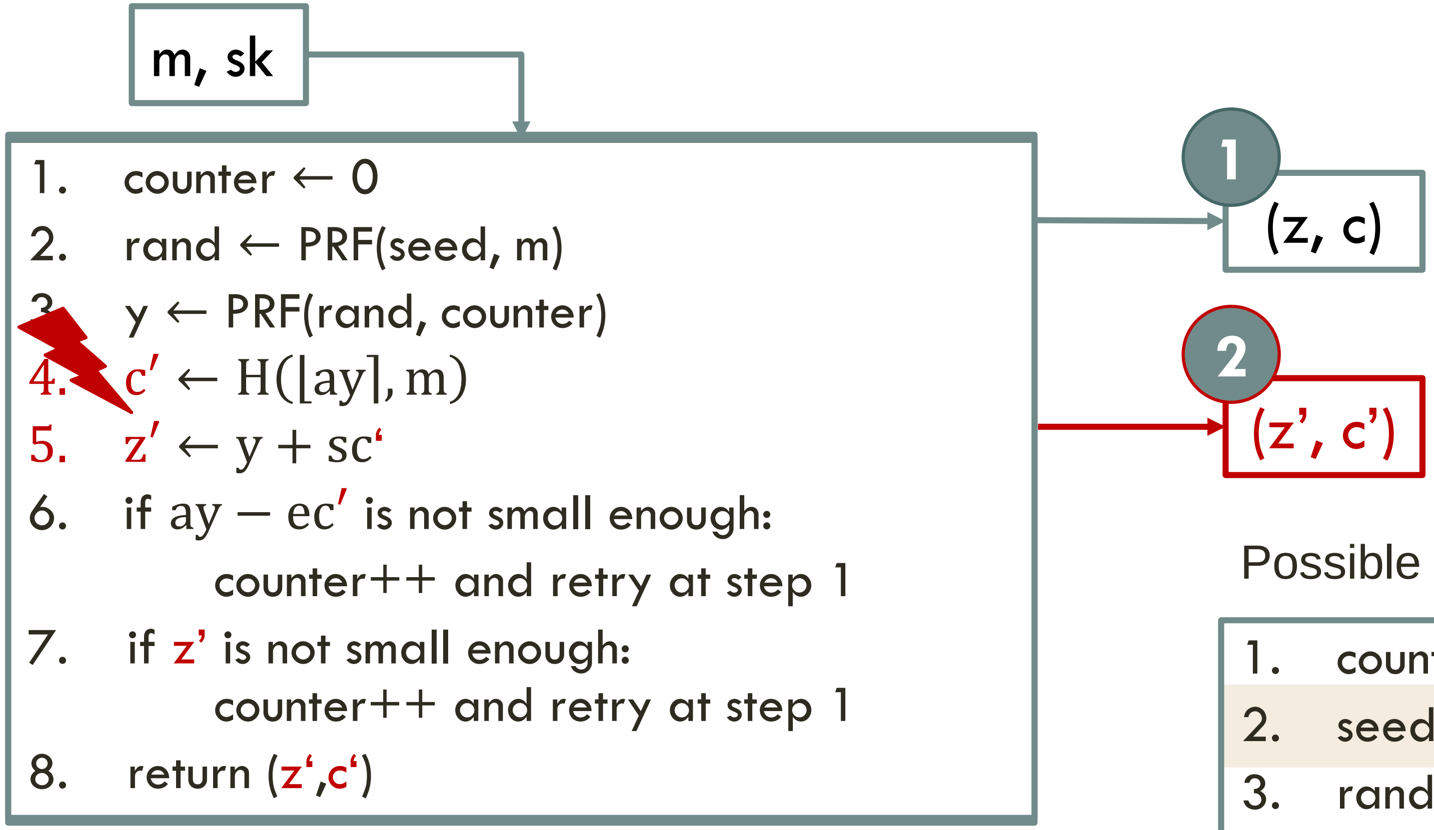
```
1. counter ← 0
2. rand ← PRF(seed, m)
3. y ← PRF(rand, counter)
4. $c \leftarrow H(\lfloor ay \rceil, m)$
5. $z \leftarrow y + sc$
6. if $ay - ec$ is not small enough:
       counter++ and retry at step 1
7. if z is not small enough:
       counter++ and retry at step 1
8. return (z,c)
```

①

(z, c)

FAULT ATTACK ON DETERMINISTC SIGNATURE

by Poddebniak, Somorovsky, Schinzel, Lochter, and Rösler [eprint 2017/1014]



m, sk

```
1.  counter ← 0
2.  rand ← PRF(seed, m)
3.  y ← PRF(rand, counter)
4.  $c' \leftarrow H(\lfloor ay \rceil, m)$
5.  $z' \leftarrow y + sc'$
6.  if $ay - ec'$ is not small enough:
        counter++ and retry at step 1
7.  if z' is not small enough:
        counter++ and retry at step 1
8.  return (z',c')
```

$z - z' = sc + y - sc' - y$

$\quad\;\; = s(c - c')$

$c - c'$ known → compute s

Possible countermeasure:

```
1.  counter ← 0
2.  seed $\leftarrow_{\$} \{0,1\}^{\kappa}$
3.  rand ← PRF(seed, m)
4.  y ← PRF(rand, counter)
```

Weak randomness enough

OUTLINE

How fault attacks shape the design

Known attacks

Probabilistic
vs.
deterministic

How side channels shape the design

Gaussian sampling

Analysis of cache side channels using program semantic

GAUSSIAN VS UNIFORM SAMPLING DURING SIGN

Signature $z = y + sc$

	Gaussian sampling of randomness		Uniform sampling of randomness used in qTESLA
✓	Small signatures	✗	Large signatures
✗	Complicated implementation of rejection sampling	✓	Easy rejection sampling
✗	Hard to implement without side channels	✓	Easy to implement without side channels

Attack on rejection sampling of BLISS
[eprint 2017/505]

Key recovery attack on BLISS and mitigations:
[eprint 2016/300, 2016/276, 2017/033, 2017/490]

OUTLINE

How fault attacks shape the design

Known attacks

Probabilistic
vs.
deterministic

How (cache-) side channels shape the design

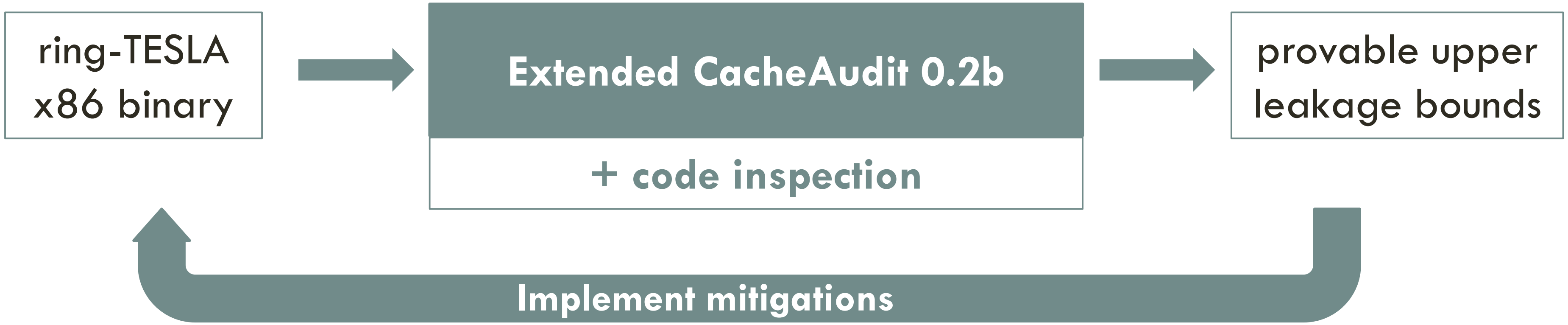
Gaussian sampling

Analysis of cache side channels using program semantic

CACHE SIDE CHANNELS

- Cache = memory to store entries for quick access
- cached entries are available faster (hit) than uncached entries (miss)
  → example attack: measure victim execution time

- Analysis of cache-side-channel vlunerability with code inspection and program analysis [eprint 2017/951]



ring-TESLA x86 binary

Extended CacheAudit 0.2b

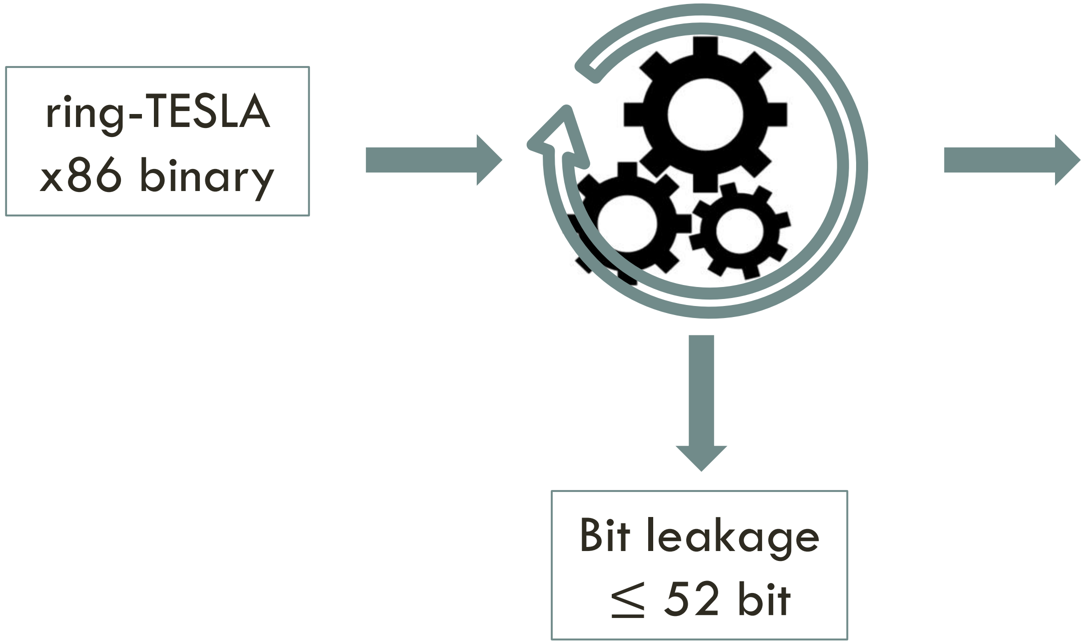
+ code inspection

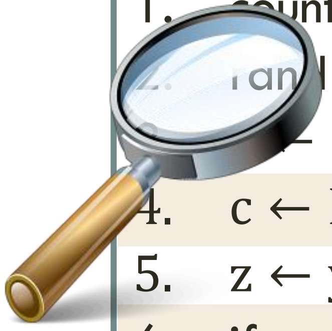
provable upper leakage bounds

Implement mitigations

INTERPRETATION OF LEAKAGE BOUNDS

- zero leakage ⇨ ✓ **provably** no cache side channel wrt to attack model
- non-zero leakage ⇨ ≥ **potential** vulnerabilities





```
1.   counter ← 0
2.   rand ← PRF(seed, m)
3.     ← PRF(rand, counter)
4.   $c \leftarrow H(\lfloor ay \rceil, m)$
5.   $z \leftarrow y + sc$
6.   if $ay - ec$ is not small enough:
         counter++ and retry at step 1
7.   if z is not small enough:
         counter++ and retry at step 1
8.   8.   return (z,c)
```

MITIGATION IN SUBROUTINES = ZERO LEAKAGE?

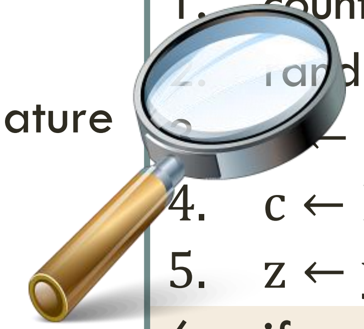
- Mitigation in subroutines does not lead to zero leakage in sign

Why?

  - length of cache trace depends on rejection
  - only leaks the number of tries to generate valid signature

- upper bounds are conservative, not tight

- bounds are low compared to key size
  - key size: 49 152 bit*
  - bit leakage: 48.6 bit* → 0.1% of bits are leaked



```
1. counter ← 0
2. rand ← PRF(seed, m)
   ← PRF(rand, counter)
4. $c \leftarrow H(\lfloor ay \rceil, m)$
5. $z \leftarrow y + \ sc$
6. if $ay - ec$ is not small enough:
       counter++ and retry at step 1
7. if z is not small enough:
       counter++ and retry at step 1
8. 8. return (z,c)
```

* results correspond to ring-TESLA;
qTESLA should be about the same

CONCLUSION

- Summarized state-of-the-art of implementation attacks for lattice-based signature schemes
- We saw that …
  - … concrete fault attack influence choice of secret key
  - … deterministic signatures might be more vulnerable to a fault attack
  - … side channels influence the choice of randomness during sign
  - … the provable mitigation of some chache side channels is very hard – even impossible – because of the design
- Disclaimer: no performance comparison

TECHNISCHE
UNIVERSITÄT
DARMSTADT

CROSSING

Special thanks to Alexandra Weber for her inspiration regarding the cache-side-channel slides!

THANKS