LATTICE-BASED SIGNATURE SCHEMES AND THEIR SENSITIVITY TO FAULT ATTACKS

FDTC 2016
08/16/2016

Nina Bindel
Johannes Buchmann
Juliane Krämer
TU Darmstadt

SHOR’S ALGORITHM 1994

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

RSA and ECDSA insecure

A digital computer is g
device; that is, it is believ
an increase in computation
true when quantum mech
factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.

Keywords: algorithmic number theory, prime factorization, discrete logarithms, Church’s thesis, quantum computers, foundations of quantum mechanics, spin systems, Fourier transforms

AMS subject classifications: 81P10, 11Y05, 68Q10, 03D10

QUANTUM COMPUTER REALISTIC?

- John Martinis (UCSB & Google Quantum Labs):
  until 2019 universal quantum computer
- Prediction by EU-commision:
  until 2035 universal quantum computer

BETTER SAFE THAN SORRY

- NSA, 2015 : announcement about transition from classical to quantum-resistant crypto
- NIST, 2016: announcement to start standardization competition

POST-QUANTUM CANDIDATES

- Quantum key distribution

- Multivariate Crypto
- Code-based Crypto
- Hash-based Crypto
- **Lattice-based Crypto**

Side-channel analysis

Fault analysis

CONTRIBUTION

- Analysis of LBSS: BLISS, GLP, ring-TESLA

- 1st order attacks
- Randomization, skipping, zeroing

- all-in-all 15 different attacks
- to 9 at least one scheme vulnerable

- Propose countermeasures

VULNERABILITIES OF LBSS

Fault Attack	Changed Value or Op.	Algorithm	GLP	BLISS	ring-TESLA
Randomization	Secret	Sign	●	●	○
Skipping	Addition	Key Gen	●	●	●
	Addition	Sign	●	○	○
	Correctness check	Verify	●	●	●
	Size check	Verify	●	●	○
Zeroing	Secret	Key Gen	●	-	○
	Randomness	Sign	●	●	●
	Hash polynomial	Sign	●	●	●

NOTATION

- $R_q = \mathbb{Z}_\mathrm{q}[\mathrm{x}]/(\mathrm{x}^\mathrm{n} + 1)$, i.e., polys of degree n-1 with coefficients in $\left[-\frac{\mathrm{q}}{2}, \frac{\mathrm{q}}{2}\right]$
- Security assumption: Learning with errors (R-LWE)

  Short integer solution (R-SIS)

LATTICE-BASED HARDNESS ASSUMPTION

R-LWE

Secret key

Public key

a · s + e = b mod q

$a \xleftarrow{\$} R_q$

$s_i, e_i \leftarrow D_\sigma$ or “small”

IDEA RANDOMIZATION ATTACK

- Based on Bao et al. [BDHJNN96]

Fault injection:

Change coefficient of original secret

Software computation:

Find index and value of faulted secret

DESCRIPTION KEY GENERATION OF GLP SCHEME

Key Generation Input: 1^{κ} Output: pk, sk	1. $\mathrm{s}, \mathrm{e} \leftarrow$ poly with coeffs $\in \{-1,0,1\}$ 2. $\mathrm{a} \underset{\$}{\leftarrow} \mathbb{Z}_{\mathrm{q}}[\mathrm{x}]/(\mathrm{x}^{\mathrm{n}} + 1)$ 3. $\mathrm{b} \leftarrow \mathrm{as} + \mathrm{e} \bmod \mathrm{q}$ 4. $\mathrm{sk} = \mathrm{s}, \mathrm{pk} = (\mathrm{a}, \mathrm{b})$ 5. Return $(\mathrm{pk}, \mathrm{sk})$

DESCRIPTION OF GLP SCHEME

Signature Generation Input: $sk = (s, e), \mu$ Output: $\sigma = (z_1, z_2, c)$	1. $y_1, y_2 \leftarrow \$$ 2. $c \leftarrow H(ay_1 + y_2, \mu)$ 3. $z_1 \leftarrow y_1 + sc$ 4. $z_2 \leftarrow y_2 + ec$ 5. Return (z_1, z_2, c) with some probability
Verification Input: $\sigma, \mu, pk = (a, b)$ Output: $\{0,1\}$	1. Check size of z_1, z_2 2. Check $c = H(az_1 + z_2 - bc, \mu)$ 3. If both checks okay: accept 4. Otherwise: reject

STRUCTURE ATTACK

Assumption 1

1st Insert fault: change one coeff. $\mathrm{s_i} \in \{-1,0,1\}$ to $\mathrm{s_i}' \in \{-1,0,1\}$

Assumption 2:
coeffs. saved in 2 bit

2nd Software computation: find index i and determine value of $\mathrm{s_i}$

- **1st** find $\mathrm{s_i} - \mathrm{s_i}'$ at index $\mathbf{i}$
- **2nd** compute $\mathrm{s_i}$

FAULTED SIGNATURE

Signature Generation

Input: $sk = (s, e), \mu$

Output: $\sigma = (z_1, z_2, c)$

1. $y_1, y_2 \leftarrow \$$
2. $c \leftarrow H(ay_1 + y_2, \mu)$
3. $z_1 \leftarrow y_1 + s'c$
4. $z_2 \leftarrow y_2 + ec$
5. Return (z_1, z_2, c) with some probability

During verification check $c = H(az_1 + z_2 - bc, \mu)$

Instead check $c = H(az_1 + z_2 - bc - a\alpha x^i c, \mu)$ for values $\alpha \in \{-2, -1, 0, 1, 2\}$ and $i \in \{0, \dots, n-1\}$

FINDING INDEX AND VALUE

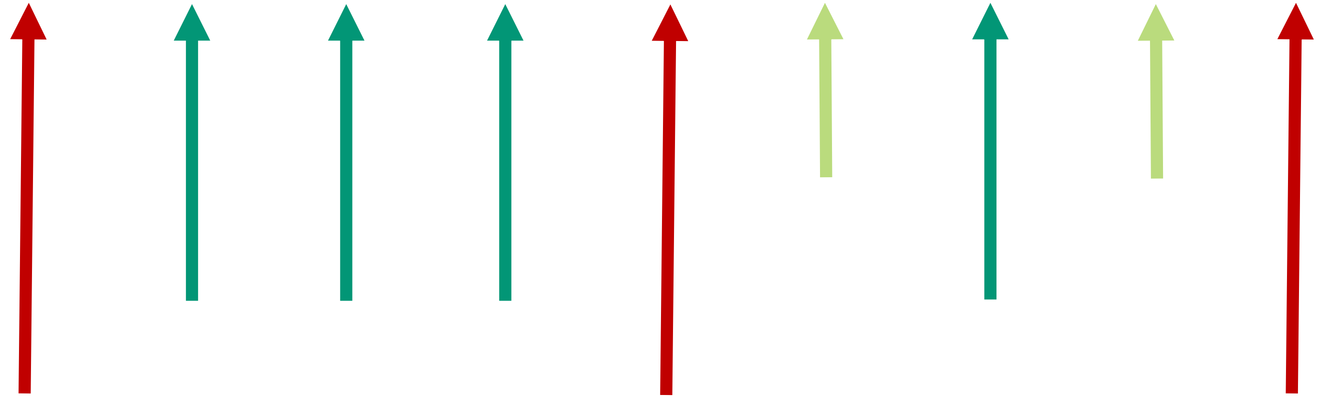
For which values $\alpha \in \{-2, -1, 0, 1, 2\}$ and $i \in \{0, \ldots, n-1\}$ does the equation …

$$
\begin{aligned}
c &= H\left(az_1 + z_2 - bc - a\alpha x^i c, \mu\right) \\
&= H\left(a(y_1 + s'c) + y_2 + ec \; - (as + e)c \; - a\alpha x^i c, \mu\right) \\
&= H\left(ay_1 + y_2 + a(s' - s - \alpha x^i)c, \mu\right)
\end{aligned}
$$

… hold?

DETERMINATION OF COEFFICIENT

s_i	0	0	0	1	1	1	-1	-1	-1
s_i'	0	1	-1	0	1	-1	0	1	-1
$\alpha = s_i - s_i'$	**0**	**-1**	**1**	**1**	**0**	**2**	**-1**	**-2**	**0**



Probability to uniquely determine coeff.: 2/9

NUMBER OF NEEDED FAULTS

Number of secret coefficients: n = 512

→ plain expected number of faults: $\frac{9}{2} \cdot 512 \approx 2304$

Reduce number of faults:

Hybrid approach of **fault attacks** and **mathematical crypanalysis of LWE**

Enough to determine **118** of the secret coefficients

→ expected number of faults: $\frac{9}{2} \cdot 118 \approx 531$

HYBRID APPROACH

- LWE gets easier when part of the secret known

- Software Computation time: 1 day
- Lattice cryptanalysis [LP10]: 118 coefficients necessary

→ Coefficients by fault attacks: 118
→ Coefficients by lattice-based cryptanalysis: 396

GENERALIZATIONS

- change more than one coefficient per fault
  - decreases number of expected faults
  - increases run time to find coefficients

- apply similar approach to BLISS
  - coeffs chosen in small interval
- not feasible for ring-TESLA
  - coeffs chosen Gaussian distributed

→ One countermeasure: use LWE with Gaussian distribution

COUNTERMEASURE

1. $y_1, y_2 \leftarrow \$$
2. $c \leftarrow H(ay_1 + y_2, \mu)$
3. $b' = as' + e \bmod q$
4. $z_1 \leftarrow a^{-1}(b - b')c + s'c + y_1$
5. $z_2 \leftarrow y_2 + ec$
6. Return (z_1, z_2, c)

Disadvantage:

- Additional computation: a^{-1}, b'
- Additional input: b

$$
\begin{aligned}
z_1 &= a^{-1}(b - b')c + s'c + y_1 \\
&= a^{-1}(as + e \ - as' - e) + s'c + y_1 \\
&= a^{-1}a(s - s')c + s'c + y_1 \\
&= sc + y_1
\end{aligned}
$$

FUTURE WORK

- implement and run attack in praxis
- implement countermeasures and evaluate their effectiveness

TECHNISCHE UNIVERSITÄT DARMSTADT

CROSSING

THANKS