Practical Lattice-based Digital Signature Schemes

NIST Workshop on Cybersecurity in a Post-Quantum World

J. Howe, T. Pöppelmann, M. O’Neill, E. O’Sullivan,
T. Güneysu, V. Lyubashevsky

Lattice-Based Cryptography

- **Why focus on lattice-based cryptography?**
  - Solid theoretical foundation and problems (CVP, SVP, SIS, LWE)
  - More versatile than code-based, MQ, and hash-based schemes:
    → Can realize signature **and** encryption schemes
    → Supports advanced constructions (e.g., IBE, ABE, FHE)
  - First evidence for the efficiency of schemes in practice

Challenges for (Lattice) Cryptography in Practice

- **Challenges for Next-Gen Cryptography**
  - As efficient and versatile as classical PK-systems, such as RSA and ECC
  - Embedded devices are constrained
    - No large memories
    - Limited computational power
  - Choice of parameters is crucial
    - Directly affects performance
    - Long-term/QC-security
    - Scalability and performance impact

- **Key Requirements**
  - Efficient/inexpensive both in HW & SW
  - Small keys, ciphertexts, signatures
  - Resistance against quantum computers and physical attacks

SAFEcrypto

Foundations of Lattice-Based Cryptography

- **General lattices** come with solid security guarantees from worst-to-average case security reduction but **are large and lack efficiency**

- **Ideal lattices** introduces algebraic structure into previously random lattices with no serious advantage for attackers so far
  - Ideals in the ring $\mathrm{R} = Z_q[x]/\langle x^n + 1\rangle$ with n being a power of two and q being a prime such that $q = 1 \bmod 2n$ (*)
  - Most standard lattice problems have an ideal lattice counterpart

- Popular problems for cryptography are the **Shortest Integer Solution (SIS)** and **Learning With Error (LWE)** problem

- NTRUEncrypt exists since 1996 with no significant attacks to date.

(*) Though other choices for parameters are possible, too, these parameters have emerged as a good compromise regarding security and efficiency.

Lattice-Based Signatures and Implementation Efficiency

- Hash-and-Sign Signatures
  - ~~NTRUSign [Hoffstein et al. 2003]~~ — Broken
  - Fixed NTRUSign [Melchor et al. 2014] — Efficient in SW
  - GPV [Gentry et al. 2008] — Less efficient
  - DLP [Ducas et al. 2014] — Efficient in SW

- Fiat-Shamir Signatures
  - LYU [Lyubashevsky 2012] — Less efficient
  - PASSSign [Hoffstein et al. 2014] — Efficient in SW
  - GLP [Güneysu et al. 2012] — Efficient in SW and HW
  - BLISS [Ducas et al. 2013] — Efficient in SW and HW
  - BG [Bai and Galbraith 2014] — Under review

Note: These statements reflect the current assessment of costs and efficiency based on existing/projected implementations. May be subject to change.

SAFEcrypto

Fiat-Shamir Signature Schemes [Lyu09, Lyu12, DDLL13]

Secret Key: **S** $\in \mathbb{Z}_q^{m\times k}$, short

Public Key: (**A**,**T**), where **A** $\in \mathbb{Z}_q^{n\times m}$ and **T**=**AS** mod *q*

Sign(*μ*)

Pick a random **y** $\leftarrow D_\sigma^m$, short

Compute **c**=H(**Ay** mod *q*,μ)

z=**Sc**+**y**

Output(**z**,**c**) with probability

min (D_σ^m(**z**) / *M.* $D_{Sc,\sigma}^m$(**z**) , 1)

Verify(***z***,***c***)

Check that ‖**z**‖ is "small"

and

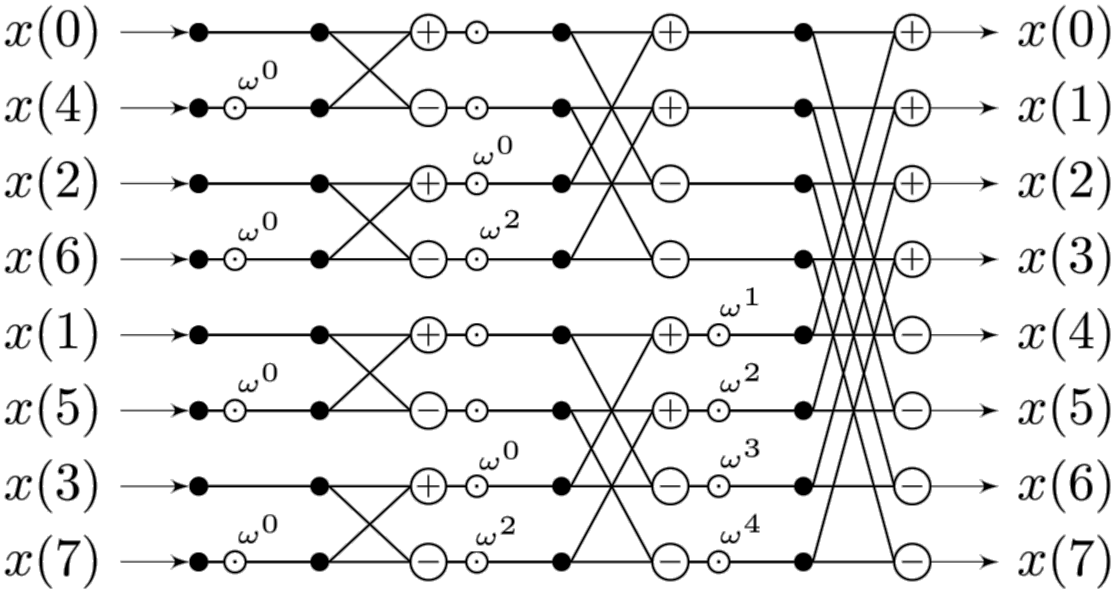
c = H(**Az** – **Tc** mod *q*, *μ*)

Components and Implementation Challenges

- **Ingredients for Fiat-Shamir-based signature scheme**
  - **Polynomial multiplication**
    - Runtime $O(n \log(n))$ when using the Number Theoretic Transform (NTT)
    - Requires transformation of parameters to/from NTT domain
    - Compute sequence $a * b = \text{INTT}\big(\text{NTT}(a) \circ \text{NTT}(b)\big)$ with $a, b \in R$
  - **Discrete Gaussian sampling (A)**
    - Some schemes require high precision for Gaussian samplers
    - Complex exponential function evaluation or large sampling tables
    - Sampling process should not be a bottleneck (can be parallelized)
  - **Discrete uniform sampling (B)**
    - Technically simpler to implement than Gaussian sampling
    - Leads to larger signatures

Implementation of the Number-Theoretic Transform (NTT)

- **Polynomial multiplication is crucial for overall performance**
- Cooley-Tukey decimation-in-time NTT algorithm requires bit-reversal and $\frac{n}{2}\log_2(n)$ multiplications in Z_q

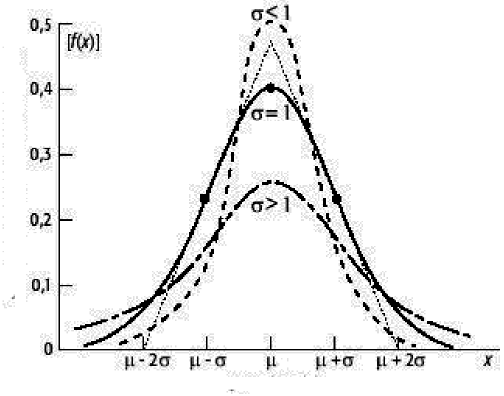


- Trick: Keep/store parameters in NTT representation if possible
- For GLP parameter set I: 4480 cycles on Core i5-3210M CPU

SAFEcrypto

How to implement Gaussian Sampling

- **Task**: avoid large tables and costly evaluation of exp. function
- **Proposed sampling techniques**
  - Rejection sampling (straight, expensive)
  - Bernoulli (quite efficient and fast)
  - Discrete Ziggurat (moderately fast)
  - Knuth-Yao (moderately large tables)

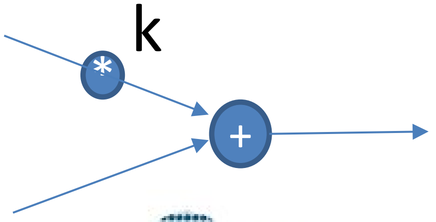


- **State of the art**: Cumulative Distribution Tables [PDG14]
  - Convolution theorem to combine values from smaller tables
  - Implement guide table to accelerate sampling process

```
0 -> 0x55,0xd9,0xc4,0x9d,0x20,0x62
1 -> 0x87,0xef,0x8a,0xd2,0x36,0x65
2 -> 0x0f,0x09,0x3c,0xed,0xf2,0x36
3 -> 0x00,0x0d,0x59,0x49,0xaf,0x8e
4 -> 0x00,0x00,0x02,0x1d,0x57,0x70
5 -> 0x00,0x00,0xa5,0x68,0x24,0xbf
6 -> 0x00,0x00,0xe1,0x2b,0x2f,0x90
7 -> 0x00,0x00,0xf5,0xfe,0x6d,0x8a
8 -> 0x00,0x00,0xfc,0xe7,0x4e,0x4e
9 -> 0x00,0x00,0x00,0x16,0x20,0x75
```

```
0 -> 0x55,0xd9,0xc4,0x9d,0x20,0x62
1 -> 0x87,0xef,0x8a,0xd2,0x36,0x65
2 -> 0x0f,0x09,0x3c,0xed,0xf2,0x36
3 -> 0x00,0x0d,0x59,0x49,0xaf,0x8e
```

```
0 -> 0x55,0xd9,0xc4,0x9d,0x20,0x62
1 -> 0x87,0xef,0x8a,0xd2,0x36,0x65
2 -> 0x0f,0x09,0x3c,0xed,0xf2,0x36
3 -> 0x00,0x0d,0x59,0x49,0xaf,0x8e
```



Implementing Lattice-Based Signature Schemes: Progress

- **Fiat-Shamir schemes BLISS and GLP received most attention**
  High performance implementation on AVR, ARM, FPGA, and PC
  - High security levels and short signatures/keys
  - Linear impact on performance when scaling parameters

- **Open implementation issues and research questions**
  - Low-cost implementation on ASIC/RFID
  - Vulnerability against physical attacks & countermeasures

- **Further steps and standardization**
  - Lattice-based constructions are efficient and highly versatile
  - High-performance **and** long-term security
  - Practical lattice-based cryptography is still young
    → further cryptanalysis and refinement essential

SAFEcrypto

Results on Lattice-Based Signatures in SW

Scheme	Security	Sign. Size	*sk* Size	*pk* Size	Sign./s	Ver./s
GLP-I	80 bits	9.5kb	2kb	12kb	5,300	75,500
BLISS-I	128 bits	5.6kb	2kb	7kb	8,000	33,000
BLISS-II	128 bits	5kb	2kb	7kb	2,000	33,000
BLISS-III	160 bits	6kb	3kb	7kb	5,000	32,000
BLISS-IV	192 bits	6.5kb	3kb	7kb	2,500	31,000
RSA-2048	112-bits	2 kb	2 kb	2 kb	800	27,000
RSA-4096	128-bits	4 kb	4 kb	4 kb	100	7,500
ECDSA-256	128-bits	0.5 kb	0.25 kb	0.25 kb	9,500	2,500
ECDSA-384	192-bits	0.75 kb	0.37 kb	0.37 kb	5,000	100

Computing platforms:

BLISS+RSA+ECDSA; “Intel Core i7 at 3.4 GHz”, 32GB RAM with OpenSSL 1.0.1c [DDLL13]

GLP-I: Intel Core i5-3210M at 3.4 GHz, based on cycle counts [GOPS14]

Results on Lattice-Based Signatures in HW

Scheme	Security	Description	Device	Resources	Ops/s
GLP-I (Sign)	80-bits	$q = 8383489, n = 512$	S6 LX16	7,465 LUT/ 8,993 FF/ 28 DSP/ 29.5 BRAM18	931
GLP-I (Ver)	80-bits	$q = 8383489, n = 512$	S6 LX16	6,225 LUT/ 6,663 FF/ 8 DSP/ 15 BRAM18	998
BLISS-I (Sign)	128-bits	CDT sampler	S6 LX25	7,491 LUT/ 7,033 FF/ 6 DSP/ 7.5 BRAM18	7,958
BLISS-I (Sign)	128-bits	Bernoulli sampler	S6 LX25	9,029 LUT/ 8,562 FF/ 8 DSP/ 6.5 BRAM18	8,081
BLISS-I (Ver)	128-bits	-	S6 LX25	5,275 LUT/ 4,488 FF/ 3 DSP/ 4.5 BRAM18	14,438
RSA (Sign)	103-bits	RSA-2048; private key	V5 LX30	3,237 LS/ 17 DSPs	89
ECDSA (Sign)	128-bits	Full ECDSA; secp256r1	V5 LX110	32,299 LUT/FF pairs	139
ECDSA (Ver)	128-bits	Full ECDSA; secp256r1	V5 LX110	32,299 LUT/FF pairs	110

Results obtained on Xilinx Spartan-6 (S6) and Xilinx Virtex-6 (V6) FPGAs

Conclusion

- **Fiat-Shamir schemes are well understood and several efficient implementations for (embedded) platforms are available**
- No serious theoretical attacks on Fiat-Shamir signature schemes
- **Early adoption**: VPN solution *strongSwan* supports BLISS signature and NTRU encryption as post-quantum mode.
- Physical attacks are not evaluated yet (timing, SCA, FIA)
- Highly interesting candidate for standardization

Horizon 2020 SAFECrypto Project:
Advancing lattice-based cryptography
In theory and practice (2015-2018)