Lattice-based digital signature scheme: qTESLA

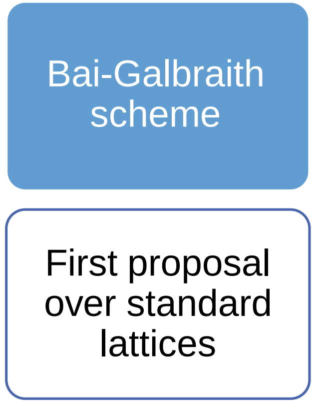
Sedat Akleylek	Ondokuz Mayis University, Turkey
Erdem Alkim	
Paulo S. L. M. Barreto	University of Washington Tacoma, USA
Nina Bindel	TU Darmstadt, Germany
Johannes Buchmann	TU Darmstadt, Germany
Edward Eaton	ISARA Corporation, Canada
Gus Gutoski	ISARA Corporation, Canada
Juliane Krämer	TU Darmstadt, Germany
Patrick Longa	Microsoft Research, USA
Harun Polat	TU Darmstadt, Germany
Jefferson E. Ricardini	**University of São Paulo, Brazil**
Gustavo Zanon	University of São Paulo, Brazil

Introduction

- qTESLA is a family of lattice-based signature schemes
- Based on decisional ring-LWE problem
- The result of a long line of research (selected):

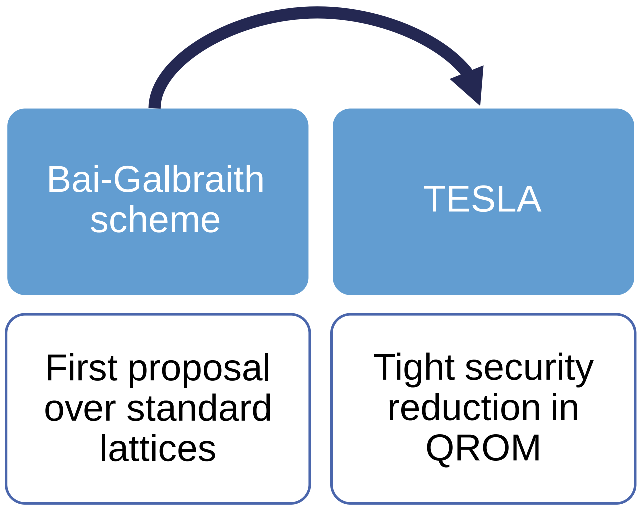
Introduction

- qTESLA is a family of lattice-based signature schemes
- Based on decisional ring-LWE problem
- The result of a long line of research (selected):



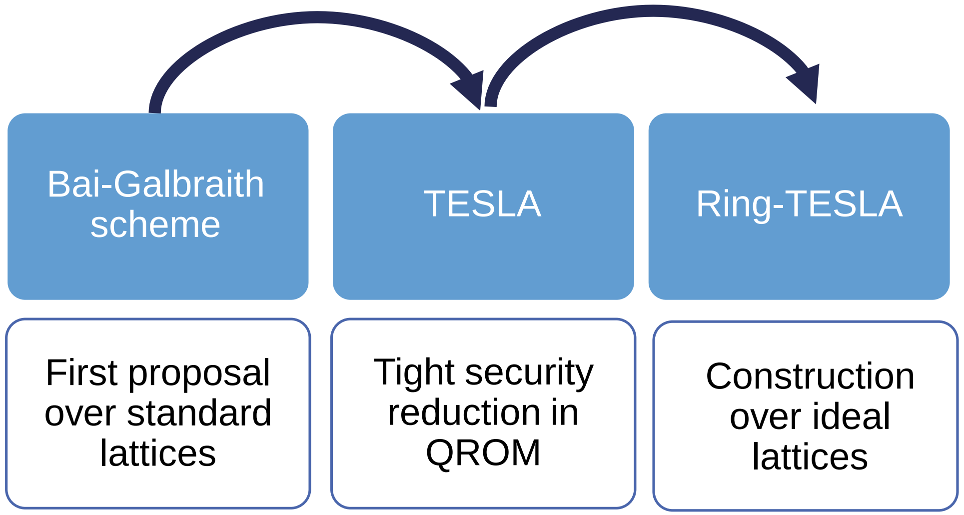
Introduction

- qTESLA is a family of lattice-based signature schemes
- Based on decisional ring-LWE problem
- The result of a long line of research (selected):



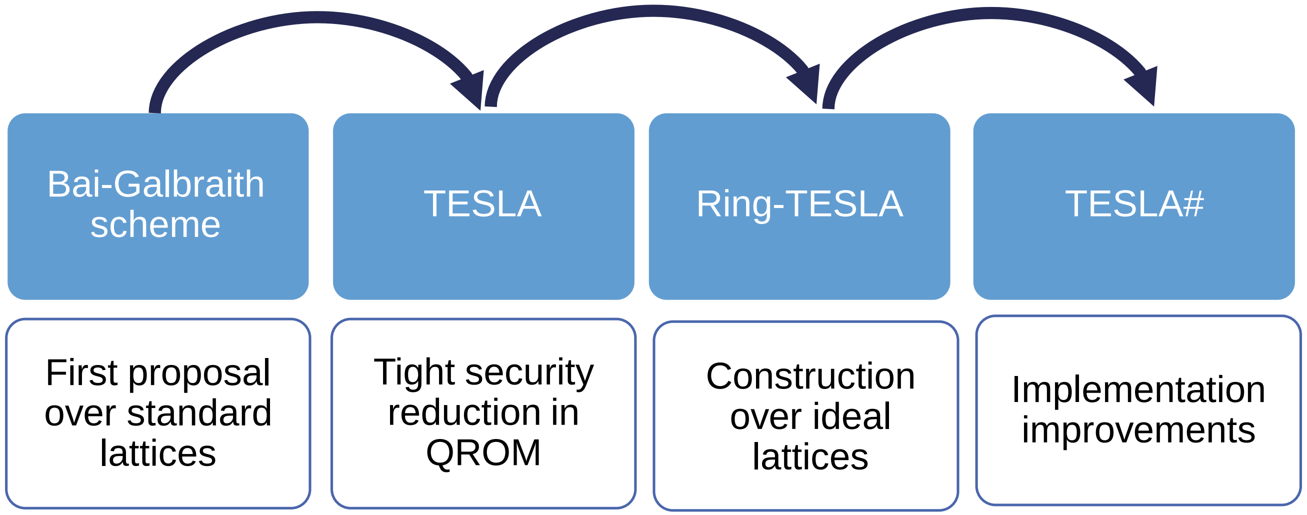
Introduction

- qTESLA is a family of lattice-based signature schemes
- Based on decisional ring-LWE problem
- The result of a long line of research (selected):



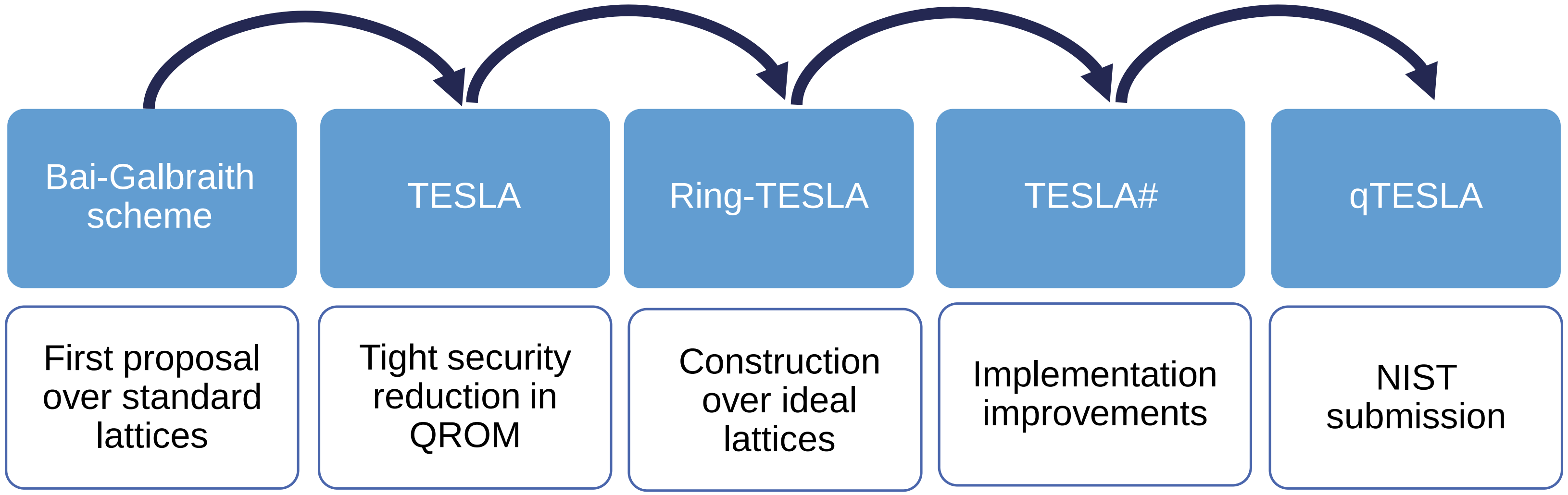
Introduction

- qTESLA is a family of lattice-based signature schemes
- Based on decisional ring-LWE problem
- The result of a long line of research (selected):



Introduction

- qTESLA is a family of lattice-based signature schemes
- Based on decisional ring-LWE problem
- The result of a long line of research (selected):



qTESLA – Key generation

- Secret key:
  - $s, e_1, \dots, e_k \overset{\sigma}{\leftarrow} \mathbb{Z}[x]/\langle x^n + 1\rangle$, “small enough”
  - $seed_a, seed_y$
- Public key:
  - $t_1 \leftarrow a_1 s + e_1 \bmod q, \dots, t_k \leftarrow a_k s + e_k \bmod q$ with $a_1, \dots, a_k \leftarrow GenA(seed_a)$
  - $seed_a$

qTESLA – Signing

Require: message m, and secret key $sk = (s, e_1, ..., e_k, \text{seed}_a, \text{seed}_y)$
Ensure: signature (c', z)

1: counter $\leftarrow 0$
2: rand $\leftarrow \text{PRF}_1(\text{seed}_y, m)$
3: $y \leftarrow \text{PRF}_2(\text{rand}, \text{counter})$
4: $a_1, ..., a_k \leftarrow \text{GenA}(\text{seed}_a)$
5: **for** $i = 1, ..., k$ **do**
6: $\quad v_i = a_i y \mod q$
7: **end for**
8: $c' \leftarrow H([v_1]_M, ..., [v_k]_M, m)$
9: $c \leftarrow \text{Enc}(c')$
10: $z \leftarrow y + sc$
11: **if** $z \notin \mathcal{R}_{q,[B-L_S]}$ **then**
12: $\quad$ counter++
13: $\quad$ Restart at step 3
14: **end if**
15: **for** $i = 1, ..., k$ **do**
16: $\quad w_i \leftarrow v_i - e_i c \mod q$
17: $\quad$ **if** $\|[w_i]_L\|_\infty > 2^d - L_E \vee \|w_i\|_\infty > \lfloor q/2 \rfloor - L_E$ **then**
18: $\quad\quad$ counter++
19: $\quad\quad$ Restart at step 3
20: $\quad$ **end if**
21: **end for**
22: **return** (c', z)

qTESLA – Signing

Require: message m, and secret key $sk = (s, e_1, ..., e_k, \text{seed}_a, \text{seed}_y)$
Ensure: signature (c', z)

1: counter $\leftarrow 0$
2: rand $\leftarrow \text{PRF}_1(\text{seed}_y, m)$
3: $y \leftarrow \text{PRF}_2(\text{rand}, \text{counter})$
4: $a_1, ..., a_k \leftarrow \text{GenA}(\text{seed}_a)$
5: **for** $i = 1, ..., k$ **do**
6: $\quad v_i = a_i y \mod q$
7: **end for**
8: $c' \leftarrow H([v_1]_M, ..., [v_k]_M, m)$
9: $c \leftarrow \text{Enc}(c')$
10: $z \leftarrow y + sc$
11: **if** $z \notin \mathcal{R}_{q,[B-L_S]}$ **then**
12: $\quad$ counter++
13: $\quad$ Restart at step 3
14: **end if**
15: **for** $i = 1, ..., k$ **do**
16: $\quad w_i \leftarrow v_i - e_i c \mod q$
17: $\quad$ **if** $\|[w_i]_L\|_\infty > 2^d - L_E \vee \|w_i\|_\infty > \lfloor q/2 \rceil - L_E$ **then**
18: $\quad\quad$ counter++
19: $\quad\quad$ Restart at step 3
20: $\quad$ **end if**
21: **end for**
22: **return** (c', z)

Pseudo-randomness expansion

qTESLA – Signing

Require: message m, and secret key $sk = (s, e_1, ..., e_k, \text{seed}_a, \text{seed}_y)$
Ensure: signature (c', z)

1: counter $\leftarrow 0$
2: rand $\leftarrow \text{PRF}_1(\text{seed}_y, m)$
3: $y \leftarrow \text{PRF}_2(\text{rand}, \text{counter})$
4: $a_1, ..., a_k \leftarrow \text{GenA}(\text{seed}_a)$

Pseudo-randomness expansion

5: **for** $i = 1, ..., k$ **do**
6: $\quad v_i = a_i y \mod q$
7: **end for**
8: $c' \leftarrow H([v_1]_M, ..., [v_k]_M, m)$
9: $c \leftarrow \text{Enc}(c')$
10: $z \leftarrow y + sc$

Computing sparse polynomial c and candidate signature z

11: **if** $z \notin \mathcal{R}_{q,[B-L_S]}$ **then**
12: $\quad$ counter++
13: $\quad$ Restart at step 3
14: **end if**
15: **for** $i = 1, ..., k$ **do**
16: $\quad w_i \leftarrow v_i - e_i c \mod q$
17: $\quad$ **if** $\|[w_i]_L\|_\infty > 2^d - L_E \vee \|w_i\|_\infty > \lfloor q/2 \rfloor - L_E$ **then**
18: $\quad\quad$ counter++
19: $\quad\quad$ Restart at step 3
20: $\quad$ **end if**
21: **end for**
22: **return** (c', z)

qTESLA – Signing

Require: message m, and secret key $sk = (s, e_1, ..., e_k, \text{seed}_a, \text{seed}_y)$
Ensure: signature (c', z)

1: counter $\leftarrow 0$
2: rand $\leftarrow \text{PRF}_1(\text{seed}_y, m)$
3: $y \leftarrow \text{PRF}_2(\text{rand}, \text{counter})$
4: $a_1, ..., a_k \leftarrow \text{GenA}(\text{seed}_a)$

Pseudo-randomness expansion

5: **for** $i = 1, ..., k$ **do**
6: $\quad v_i = a_i y \mod q$
7: **end for**
8: $c' \leftarrow H([v_1]_M, ..., [v_k]_M, m)$
9: $c \leftarrow \text{Enc}(c')$
10: $z \leftarrow y + sc$

Computing sparse polynomial c and candidate signature z

11: **if** $z \notin \mathcal{R}_{q,[B-L_S]}$ **then**
12: $\quad$ counter++
13: $\quad$ Restart at step 3
14: **end if**

“security check“ = rejection sampling

15: **for** $i = 1, ..., k$ **do**
16: $\quad w_i \leftarrow v_i - e_i c \mod q$
17: $\quad$ **if** $\|[w_i]_L\|_\infty > 2^d - L_E \vee \|w_i\|_\infty > \lfloor q/2 \rceil - L_E$ **then**
18: $\quad\quad$ counter++
19: $\quad\quad$ Restart at step 3
20: $\quad$ **end if**
21: **end for**
22: **return** (c', z)

qTESLA – Signing

Require: message m, and secret key $sk = (s, e_1, ..., e_k, \text{seed}_a, \text{seed}_y)$

Ensure: signature (c', z)

1: $\text{counter} \leftarrow 0$
2: $\text{rand} \leftarrow \text{PRF}_1(\text{seed}_y, m)$
3: $y \leftarrow \text{PRF}_2(\text{rand}, \text{counter})$
4: $a_1, ..., a_k \leftarrow \text{GenA}(\text{seed}_a)$

Pseudo-randomness expansion

5: **for** $i = 1, ..., k$ **do**
6: $\quad v_i = a_i y \mod q$
7: **end for**
8: $c' \leftarrow H([v_1]_M, ..., [v_k]_M, m)$
9: $c \leftarrow \text{Enc}(c')$
10: $z \leftarrow y + sc$

Computing sparse polynomial c and candidate signature z

11: **if** $z \notin \mathcal{R}_{q,[B-L_S]}$ **then**
12: $\quad \text{counter}{+}{+}$
13: $\quad$ Restart at step 3
14: **end if**

“security check“ = rejection sampling

15: **for** $i = 1, ..., k$ **do**
16: $\quad w_i \leftarrow v_i - e_i c \mod q$
17: $\quad$ **if** $\|[w_i]_L\|_\infty > 2^d - L_E \vee \|w_i\|_\infty > \lfloor q/2 \rfloor - L_E$ **then**
18: $\quad\quad \text{counter}{+}{+}$
19: $\quad\quad$ Restart at step 3
20: $\quad$ **end if**
21: **end for**

“correctness check“

22: **return** (c', z)

Advantages of qTESLA

- Flexible choice of parameters
  - 2 approaches to instantiate qTESLA
    - Heuristic
    - Provable-secure
- Ease of implementation
  - Structurally simple
  - Easy-to-implement underlying arithmetic
  - Gaussian sampling is not required for signing, only for key generation
- Security against implementation attacks
  - Cache-side-channel free for signature generation
  - Isochronous Gaussian sampling and arithmetic operations
  - Countermeasures against fault attacks (by-product of checking the candidate signature)

Parameter sets

- Two parameter sets
  - Provably-secure
    - Enabling tight security reduction from R-LWE in the quantum random oracle model
    - More conservative
  - Heuristic
    - Extrapolation of the concrete complexity of solving the underlying lattice problem, without targeting a tight formal security
    - Suitable for applications where performance or signature/key sizes are the main concern

Quick background story

- Initially, exclusively provably-secure parameters
  - Choose parameters guided by the security reduction

- A mistake was found by V. Lyubashevsky (thanks!)
  - Security reduction still holds
  - Bit security estimates unchanged
  - But “provable-security” property is lost for those parameters

Potential tweaks

- Tweak 1: include “provably-secure” parameters, as originally intended
- Tweak 2: fine-tune current “heuristic” parameters to get better performance

Preliminary results

Key and signature sizes (in bytes) – NIST security category 3

Parameter set	Approach	Public key	Secret key	Signature
qTESLA-128	Heuristic	2 976	1 856	2 720
qTESLA-p-128	Provable	19 872	7 744	3 104

Performance (in kilocycles) of the reference implementation on a 2.40 GHz Intel Core i5-6300U (Skylake) processor – NIST security category 3

Parameter set	Approach	keygen	sign	verify
qTESLA-128	Heuristic	2 234	1 279	229
qTESLA-p-128	Provable	4 334	2 420	732

Coming soon!

- Updated version of the specs document
- Optimized implementations for heuristic and provable-secure parameters

Thanks!

Sedat Akleylek
Erdem Alkim
Paulo S. L. M. Barreto
Nina Bindel
Johannes Buchmann
Edward Eaton
Gus Gutoski
Juliane Krämer
Patrick Longa
Harun Polat
Jefferson E. Ricardini
Gustavo Zanon

qTESLA website:
https://tesla.informatik.tu-darmstadt.de/de/tesla

Nina says hi!