Lattice-based digital signature scheme qTESLA

(2nd NIST PQC Standardization Conference, 2019)

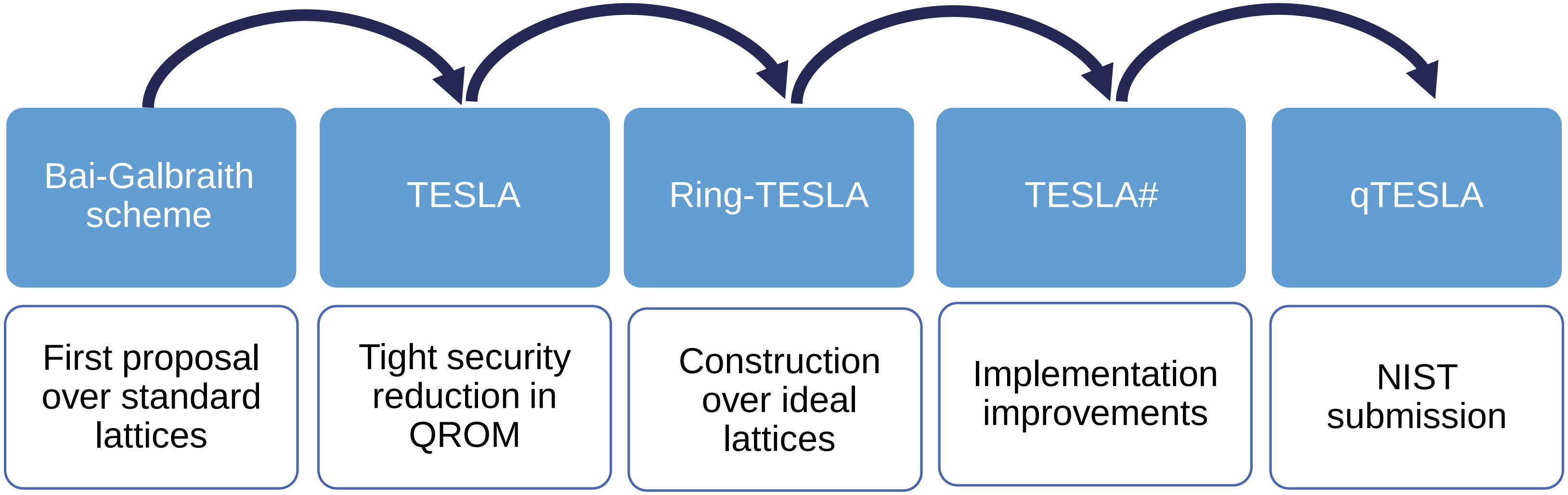
Sedat Akleylek	Ondokuz Mayis University, Turkey
Erdem Alkim	Ondokuz Mayis University, Turkey
Paulo S. L. M. Barreto	University of Washington Tacoma, USA
Nina Bindel	TU Darmstadt, Germany
Johannes Buchmann	TU Darmstadt, Germany
Edward Eaton	ISARA Corporation, Canada
Gus Gutoski	ISARA Corporation, Canada
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 post-quantum lattice-based signature schemes
- Based on the decisional R-LWE problem
- The result of a long line of research (selected):

Introduction

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



Bai-Galbraith scheme → TESLA → Ring-TESLA → TESLA# → qTESLA

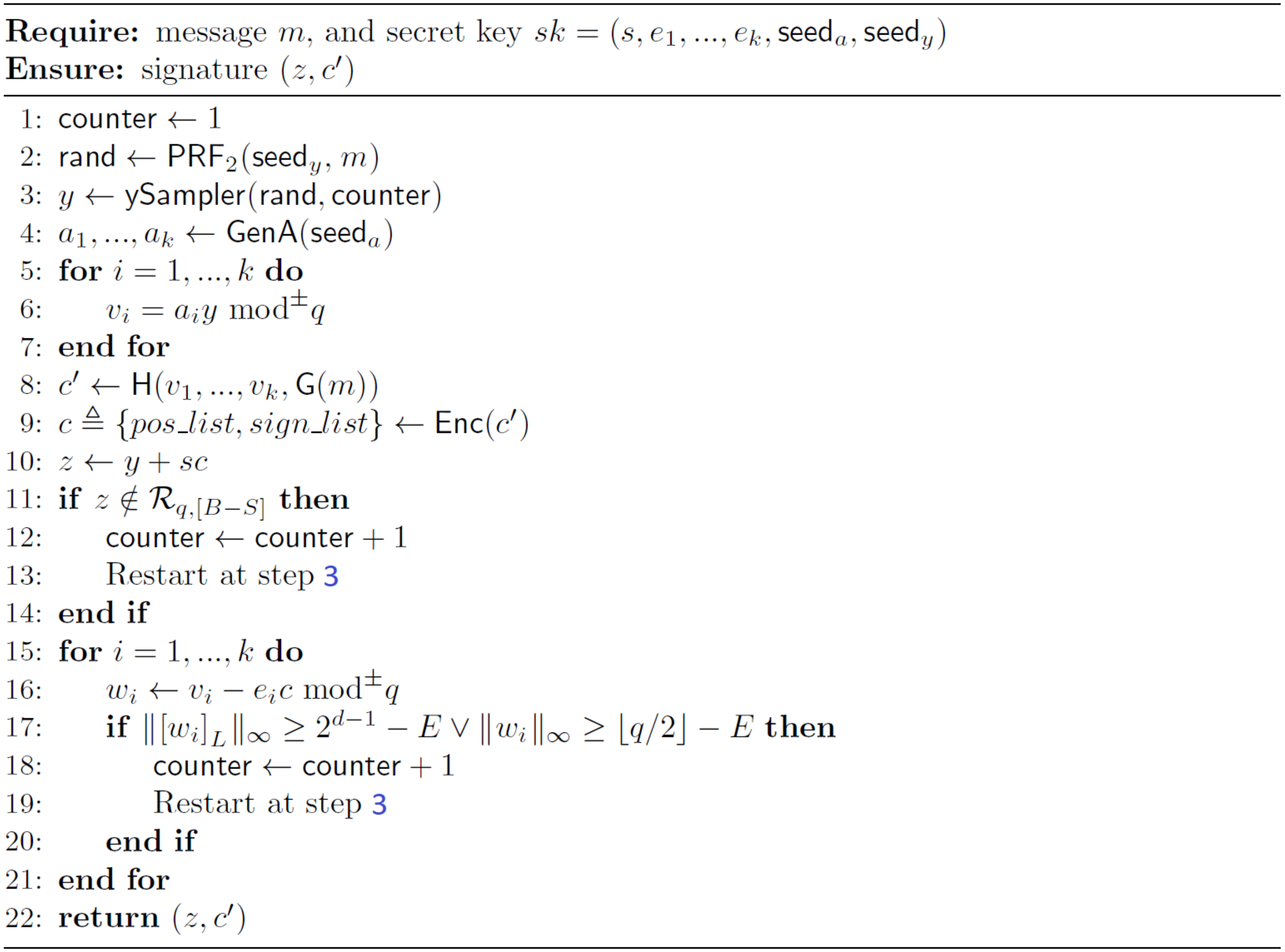
Bai-Galbraith scheme	TESLA	Ring-TESLA	TESLA#	qTESLA
First proposal over standard lattices	Tight security reduction in QROM	Construction over ideal lattices	Implementation improvements	NIST submission

qTESLA – Key generation

- Secret key:
  - $s, e_1, \dots, e_k \overset{\sigma}{\longleftarrow} \mathcal{R} = \mathbb{Z}[x]/\langle x^n + 1 \rangle$, “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 $sk = (s, e_1, ..., e_k, \mathsf{seed}_a, \mathsf{seed}_y)$
Ensure: signature (z, c')

```
 1: counter ← 1
 2: rand ← $\mathsf{PRF}_2(\mathsf{seed}_y, m)$
 3: $y$ ← ySampler(rand, counter)
 4: $a_1, ..., a_k \leftarrow \mathsf{GenA}(\mathsf{seed}_a)$
 5: for $i = 1, ..., k$ do
 6:     $v_i = a_i y \bmod^{\pm} q$
 7: end for
 8: $c' \leftarrow \mathsf{H}(v_1, ..., v_k, \mathsf{G}(m))$
 9: $c \triangleq \{pos\_list, sign\_list\} \leftarrow \mathsf{Enc}(c')$
10: $z \leftarrow y + sc$
11: if $z \notin \mathcal{R}_{q,[B-S]}$ then
12:     counter ← counter + 1
13:     Restart at step 3
14: end if
15: for $i = 1, ..., k$ do
16:     $w_i \leftarrow v_i - e_i c \bmod^{\pm} q$
17:     if $\|[w_i]_L\|_\infty \geq 2^{d-1} - E \vee \|w_i\|_\infty \geq \lfloor q/2 \rfloor - E$ then
18:         counter ← counter + 1
19:         Restart at step 3
20:     end if
21: end for
22: return $(z, c')$
```

qTESLA – Signing

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

 1: $\mathsf{counter} \leftarrow 1$
 2: $\mathsf{rand} \leftarrow \mathsf{PRF}_2(\mathsf{seed}_y, m)$
 3: $y \leftarrow \mathsf{ySampler}(\mathsf{rand}, \mathsf{counter})$
 4: $a_1, ..., a_k \leftarrow \mathsf{GenA}(\mathsf{seed}_a)$
 5: for $i = 1, ..., k$ do
 6:     $v_i = a_i y \bmod^{\pm} q$
 7: end for
 8: $c' \leftarrow \mathsf{H}(v_1, ..., v_k, \mathsf{G}(m))$
 9: $c \triangleq \{pos\_list, sign\_list\} \leftarrow \mathsf{Enc}(c')$
10: $z \leftarrow y + sc$
11: if $z \notin \mathcal{R}_{q,[B-S]}$ then
12:     $\mathsf{counter} \leftarrow \mathsf{counter} + 1$
13:     Restart at step 3
14: end if
15: for $i = 1, ..., k$ do
16:     $w_i \leftarrow v_i - e_i c \bmod^{\pm} q$
17:     if $\|[w_i]_L\|_\infty \geq 2^{d-1} - E \vee \|w_i\|_\infty \geq \lfloor q/2 \rfloor - E$ then
18:         $\mathsf{counter} \leftarrow \mathsf{counter} + 1$
19:         Restart at step 3
20:     end if
21: end for
22: return $(z, c')$
```

Pseudo-randomness expansion

qTESLA – Signing

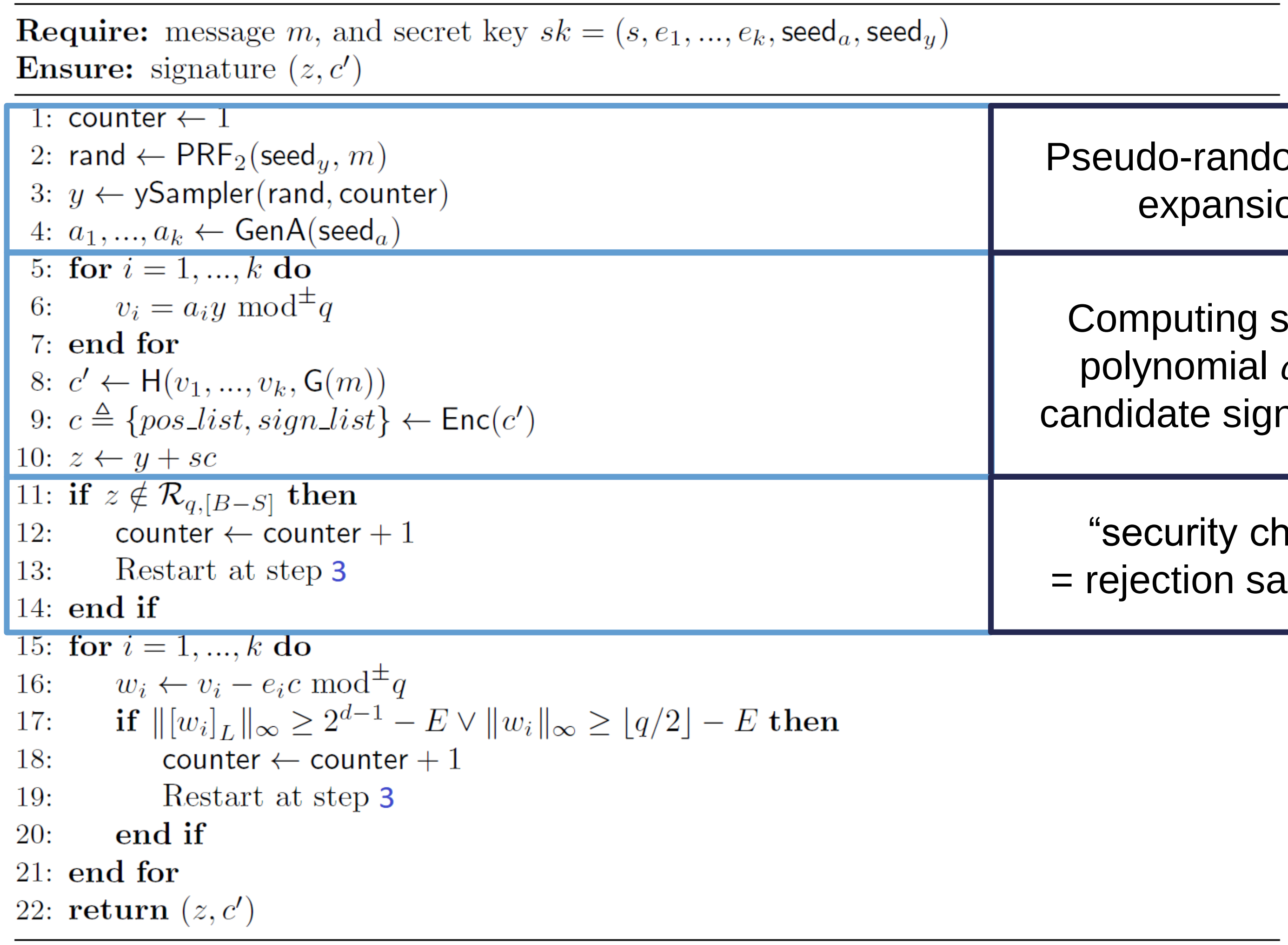
Require: message m, and secret key $sk = (s, e_1, ..., e_k, \mathsf{seed}_a, \mathsf{seed}_y)$
Ensure: signature (z, c')

```
 1: counter $\leftarrow 1$
 2: rand $\leftarrow \mathsf{PRF}_2(\mathsf{seed}_y, m)$
 3: $y \leftarrow \mathsf{ySampler}(\mathsf{rand}, \mathsf{counter})$
 4: $a_1, ..., a_k \leftarrow \mathsf{GenA}(\mathsf{seed}_a)$
 5: for $i = 1, ..., k$ do
 6:     $v_i = a_i y \bmod^{\pm} q$
 7: end for
 8: $c' \leftarrow \mathsf{H}(v_1, ..., v_k, \mathsf{G}(m))$
 9: $c \triangleq \{pos\_list, sign\_list\} \leftarrow \mathsf{Enc}(c')$
10: $z \leftarrow y + sc$
11: if $z \notin \mathcal{R}_{q,[B-S]}$ then
12:     counter $\leftarrow$ counter $+ 1$
13:     Restart at step 3
14: end if
15: for $i = 1, ..., k$ do
16:     $w_i \leftarrow v_i - e_i c \bmod^{\pm} q$
17:     if $\|[w_i]_L\|_\infty \geq 2^{d-1} - E \vee \|w_i\|_\infty \geq \lfloor q/2 \rfloor - E$ then
18:         counter $\leftarrow$ counter $+ 1$
19:         Restart at step 3
20:     end if
21: end for
22: return $(z, c')$
```

Pseudo-randomness expansion

Computing sparse polynomial c and candidate signature z

qTESLA – Signing



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

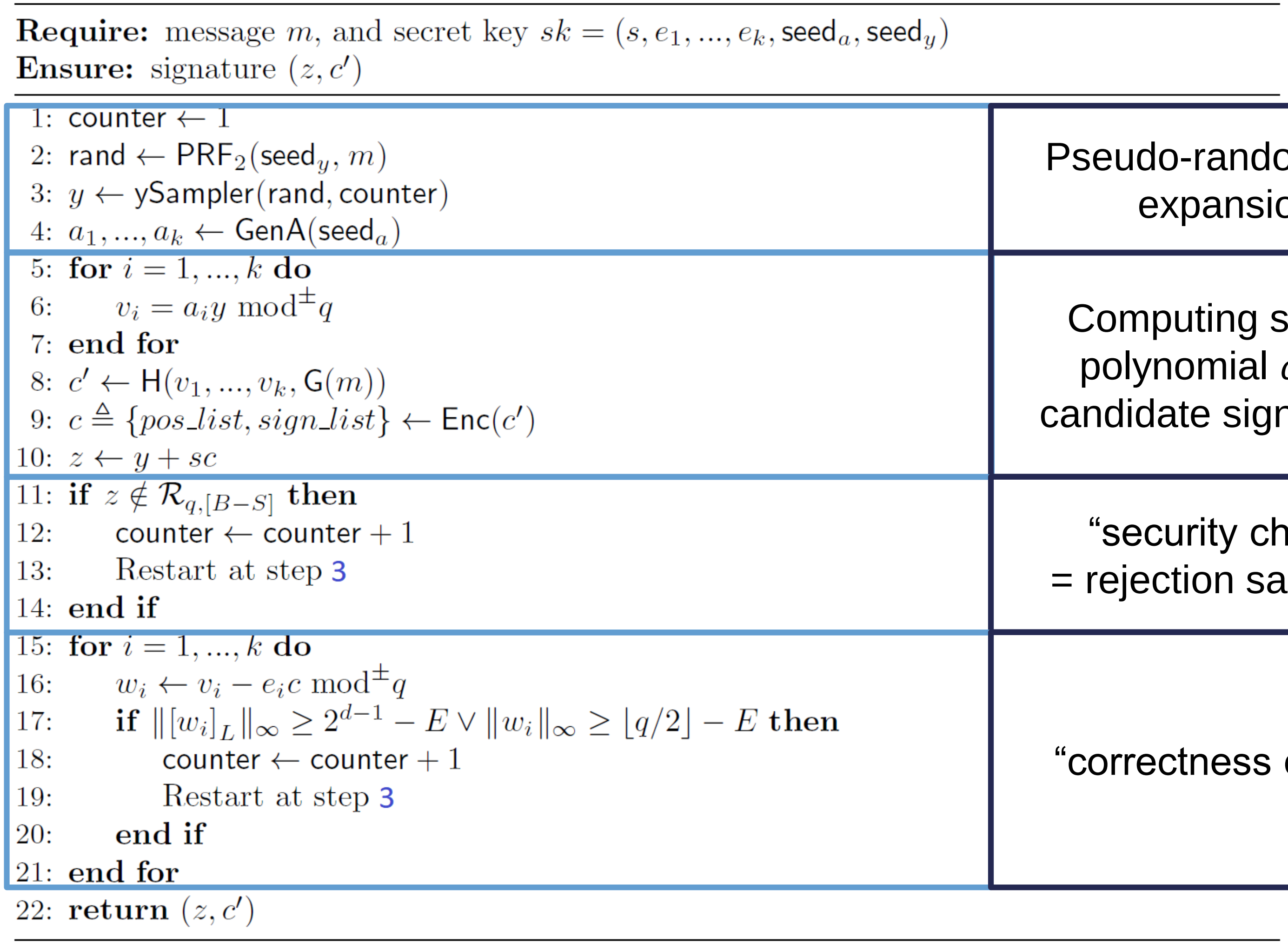
```
1:  $\mathsf{counter} \leftarrow 1$
2:  $\mathsf{rand} \leftarrow \mathsf{PRF}_2(\mathsf{seed}_y, m)$
3:  $y \leftarrow \mathsf{ySampler}(\mathsf{rand}, \mathsf{counter})$
4:  $a_1, ..., a_k \leftarrow \mathsf{GenA}(\mathsf{seed}_a)$
5:  for $i = 1, ..., k$ do
6:      $v_i = a_i y \bmod^{\pm} q$
7:  end for
8:  $c' \leftarrow \mathsf{H}(v_1, ..., v_k, \mathsf{G}(m))$
9:  $c \triangleq \{pos\_list, sign\_list\} \leftarrow \mathsf{Enc}(c')$
10: $z \leftarrow y + sc$
11: if $z \notin \mathcal{R}_{q,[B-S]}$ then
12:     $\mathsf{counter} \leftarrow \mathsf{counter} + 1$
13:     Restart at step 3
14: end if
15: for $i = 1, ..., k$ do
16:     $w_i \leftarrow v_i - e_i c \bmod^{\pm} q$
17:     if $\|[w_i]_L\|_\infty \geq 2^{d-1} - E \vee \|w_i\|_\infty \geq \lfloor q/2 \rfloor - E$ then
18:         $\mathsf{counter} \leftarrow \mathsf{counter} + 1$
19:         Restart at step 3
20:     end if
21: end for
22: return $(z, c')$
```

Pseudo-randomness expansion

Computing sparse polynomial c and candidate signature z

“security check“ = rejection sampling

qTESLA – Signing



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

```
1: counter ← 1
2: rand ← $\mathsf{PRF}_2(\mathsf{seed}_y, m)$
3: $y$ ← ySampler(rand, counter)
4: $a_1, ..., a_k \leftarrow \mathsf{GenA}(\mathsf{seed}_a)$
5: for $i = 1, ..., k$ do
6:     $v_i = a_i y \bmod^{\pm} q$
7: end for
8: $c' \leftarrow \mathsf{H}(v_1, ..., v_k, \mathsf{G}(m))$
9: $c \triangleq \{pos\_list, sign\_list\} \leftarrow \mathsf{Enc}(c')$
10: $z \leftarrow y + sc$
11: if $z \notin \mathcal{R}_{q,[B-S]}$ then
12:     counter ← counter + 1
13:     Restart at step 3
14: end if
15: for $i = 1, ..., k$ do
16:     $w_i \leftarrow v_i - e_i c \bmod^{\pm} q$
17:     if $\|[w_i]_L\|_\infty \geq 2^{d-1} - E \vee \|w_i\|_\infty \geq \lfloor q/2 \rceil - E$ then
18:         counter ← counter + 1
19:         Restart at step 3
20:     end if
21: end for
22: return $(z, c')$
```

Pseudo-randomness expansion

Computing sparse polynomial c and candidate signature z

“security check“ = rejection sampling

“correctness check“

Round 2 modifications

Round 2 modifications

Round 1	Round 2
Provably-secure parameter sets.	**Added** heuristic parameter sets.

Round 2 modifications

Round 1	Round 2
Provably-secure parameter sets.	**Added** heuristic parameter sets.
Support for power-of-two cyclotomic ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle \phi_{2^\ell}(x)\rangle$.	**Added** support for non-power-of-two cyclotomic ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle \phi_{2^\ell 9}(x)\rangle$.

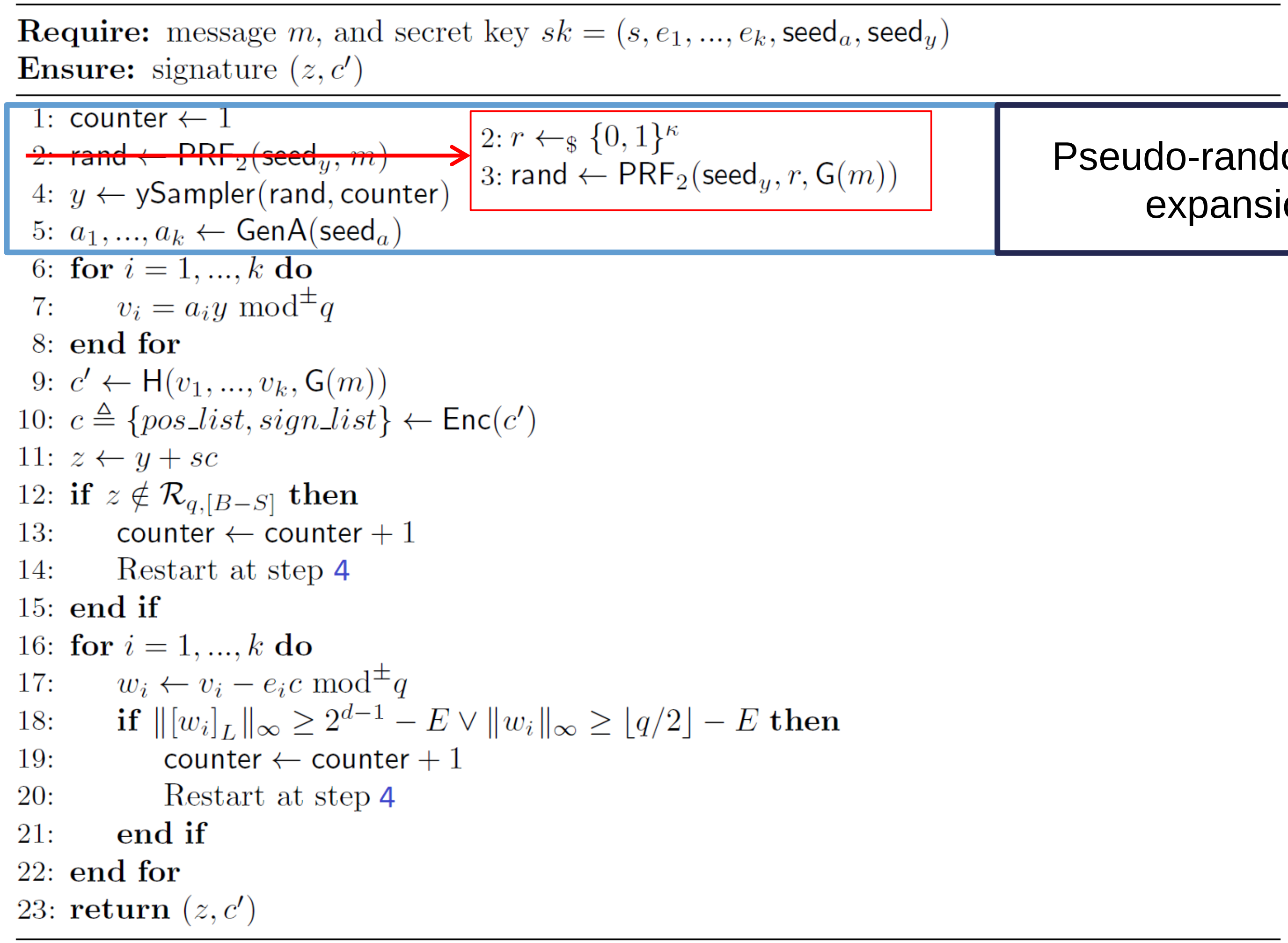
Round 2 modifications

Round 1	Round 2
Provably-secure parameter sets.	**Added** heuristic parameter sets.
Support for power-of-two cyclotomic ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle \phi_{2^\ell}(x) \rangle$.	**Added** support for non-power-of-two cyclotomic ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle \phi_{2^\ell 9}(x) \rangle$.
Simplified Bernoulli sampler: • Portability issues • Hard to make fully constant-time.	**Replaced by** simpler, faster, portable, constant-time CDT-based Gaussian sampler.

Round 2 modifications

Round 1	Round 2
Deterministic signatures.	**Converted to** probabilistic.

qTESLA – Signing



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

```
1: counter $\leftarrow 1$
~~2: rand $\leftarrow \mathsf{PRF}_2(\mathsf{seed}_y, m)$~~
    2: $r \leftarrow_{\$} \{0,1\}^{\kappa}$
    3: rand $\leftarrow \mathsf{PRF}_2(\mathsf{seed}_y, r, \mathsf{G}(m))$
4: $y \leftarrow \mathsf{ySampler}(\mathsf{rand}, \mathsf{counter})$
5: $a_1, ..., a_k \leftarrow \mathsf{GenA}(\mathsf{seed}_a)$
6: for $i = 1, ..., k$ do
7:     $v_i = a_i y \bmod^{\pm} q$
8: end for
9: $c' \leftarrow \mathsf{H}(v_1, ..., v_k, \mathsf{G}(m))$
10: $c \triangleq \{pos\_list, sign\_list\} \leftarrow \mathsf{Enc}(c')$
11: $z \leftarrow y + sc$
12: if $z \notin \mathcal{R}_{q,[B-S]}$ then
13:     counter $\leftarrow$ counter $+ 1$
14:     Restart at step 4
15: end if
16: for $i = 1, ..., k$ do
17:     $w_i \leftarrow v_i - e_i c \bmod^{\pm} q$
18:     if $\|[w_i]_L\|_\infty \geq 2^{d-1} - E \vee \|w_i\|_\infty \geq \lfloor q/2 \rfloor - E$ then
19:         counter $\leftarrow$ counter $+ 1$
20:         Restart at step 4
21:     end if
22: end for
23: return $(z, c')$
```

Pseudo-randomness expansion

Round 2 modifications

Round 1	Round 2
Deterministic signatures.	**Converted to** probabilistic.
Security reduction in the QROM using conjecture.	**Refined** conjecture and **backed it up** experimentally.

Round 2+ modifications

Round 2+ modifications

- V. Lyubashevsky pointed out that **heuristic parameters** lacked analysis of R-SIS hardness (thanks!)

Round 2+ modifications

- V. Lyubashevsky pointed out that **heuristic parameters** lacked analysis of R-SIS hardness (thanks!)
- We confirmed that R-SIS hardness was too low for round 2 heuristic parameters
- We issued an update on Aug 20, 2019

Round 2+ modifications

- V. Lyubashevsky pointed out that **heuristic parameters** lacked analysis of R-SIS hardness (thanks!)

- We confirmed that R-SIS hardness was too low for round 2 heuristic parameters

- We issued an update on Aug 20, 2019
  - A security proof following [KLS18] that reduces hardness of ST-R-SIS and R-LWE to the security of heuristic qTESLA allows generation of secure parameters
    - Main change involves increasing number of R-LWE samples from 1 to 2

[KLS18]: **A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model,** by Kiltz, Lyubashevsky, Schaffner, 2018

Round 2+ modifications

- V. Lyubashevsky pointed out that **heuristic parameters** lacked analysis of R-SIS hardness (thanks!)

- We confirmed that R-SIS hardness was too low for round 2 heuristic parameters

- We issued an update on Aug 20, 2019
  - A security proof following [KLS18] that reduces hardness of ST-R-SIS and R-LWE to the security of heuristic qTESLA allows generation of secure parameters
    - Main change involves increasing number of R-LWE samples from 1 to 2
  - However, we decided to **drop the heuristic parameters**

[KLS18]: **A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model,** by Kiltz, Lyubashevsky, Schaffner, 2018

Parameter sets

Parameter set	Heuristic			Provable	
	qTESLA-I	qTESLA-II	qTESLA-III	qTESLA-p-I	qTESLA-p-III
NIST category	1	2	3	1	3
R-LWE hardness	111	138	188	140	279
SIS hardness	50	71	95	-	-
Targeted hardness	95	128	160	95	160
pk size [bytes]	1,504	2,336	3,104	14,880	38,432
sig size [bytes]	1,376	2,144	2,848	2,592	5,664

Parameter sets

Parameter set	Heuristic			Provable	
	qTESLA-I	qTESLA-II	qTESLA-III	qTESLA-p-I	qTESLA-p-III
NIST category	1	2	3	1	3
R-LWE hardness	111	138	188	140	279
SIS hardness	50	71	95	-	-
Targeted hardness	95	128	160	95	160
pk size [bytes]	1,504	2,336	3,104	14,880	38,432
sig size [bytes]	1,376	2,144	2,848	2,592	5,664

Fixed parameter sets

Parameter set	Heuristic			Provable	
	qTESLA-I	qTESLA-II	qTESLA-III	qTESLA-p-I	qTESLA-p-III
NIST category	1	2	3	1	3
R-LWE hardness	97	130	178	140	279
SIS hardness	100	143	197	-	-
Targeted hardness	95	128	160	95	160
pk size [bytes]	2,976	4,832	6,432	14,880	38,432
sig size [bytes]	1,400	2,336	3,104	2,592	5,664

Updated parameter sets (round 2+)

Parameter set	Provable	
	qTESLA-p-I	qTESLA-p-III
NIST category	1	3
R-LWE hardness	140	279
Targeted hardness	95	160
pk size [bytes]	14,880	38,432
sig size [bytes]	2,592	5,664

Performance (round 2+)

Performance (in kilocycles) of the constant-time **reference implementation** on a 3.40GHz Intel Core i7-6700 (Skylake) processor

Parameter set	Provable	
	qTESLA-p-I	qTESLA-p-III
keygen	2,316	13,727
sign	2,325	6,285
verify	671	1,830
Total (sign + verify)	**2,996**	**8,115**

Performance (round 2+)

Performance (in kilocycles) of the constant-time **reference implementation** on a 3.40GHz Intel Core i7-6700 (Skylake) processor

Parameter set	Provable	
	qTESLA-p-I	qTESLA-p-III
keygen	2,316	13,727
sign	2,325	6,285
verify	671	1,830
Total (sign + verify)	**2,996**	**8,115**

- E.g., qTESLA-p-I produces signatures in **0.68 msec.** or **1,470 signs/sec**.

Summary of advantages

Summary of advantages

- Simple and easy to implement
  - Facilitates efficient and secure portable implementations
  - Reduces {theoretical, practical} attack surface

Summary of advantages

- Simple and easy to implement
  - Facilitates efficient and secure portable implementations
  - Reduces {theoretical, practical} attack surface
- **By default** built-in protection against some side-channel and fault attacks

Summary of advantages

- Simple and easy to implement
  - Facilitates efficient and secure portable implementations
  - Reduces {theoretical, practical} attack surface
- **By default** built-in protection against some side-channel and fault attacks
- Very conservative security
  - qTESLA instantiations are **provably-secure** in the QROM

Potential avenues of improvement

- Further optimization of implementation (e.g., using assembly).
- Use of Dilithium’s pk compression technique.

Thanks!

qTESLA website: https://qtesla.org/

Updated specs: https://qtesla.org/wp-content/uploads/2019/08/qTESLA_round2_08.19.2019.pdf

Updated package: https://qtesla.org/wp-content/uploads/2019/08/qTESLA_NIST_update_08.19.2019.zip

Code: https://github.com/qtesla/qTesla