Lattice-based digital signature scheme: qTESLA

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

Ondokuz Mayis University, Turkey
University of Washington Tacoma, USA
TU Darmstadt, Germany
TU Darmstadt, Germany
ISARA Corporation, Canada
ISARA Corporation, Canada
TU Darmstadt, Germany
Microsoft Research, USA
TU Darmstadt, Germany
University of São Paulo, Brazil
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):

- Bai-Galbraith scheme
- TESLA
- Ring-TESLA

First proposal over standard lattices
Tight security reduction in QROM
Construction over ideal lattices
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):

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

- Secret key:
  - $s, e_1, ..., e_k \leftarrow \mathbb{Z}[x]/\langle x^n + 1 \rangle$, “small enough”
  - $seed_a, seed_y$

- Public key:
  - $t_1 \leftarrow a_1s + e_1 \mod q, ..., t_k \leftarrow a_k s + e_k \mod q$ with $a_1, ..., a_k \leftarrow GenA(seed_a)$
  - $seed_a$
qTESLA – Signing

Require: message $m$, and secret key $sk = (s, e_1, ..., e_k, seed_u, seed_y)$
Ensure: signature $(c', z)$

1: counter ← 0
2: rand ← PRF_1(seed_y, m)
3: $y ← PRF_2(rand, counter)$
4: $a_1, ..., a_k ← GenA(seed_u)$
5: for $i = 1, ..., k$ do
6:   $v_i = a_i y \mod q$
7: end for
8: $c' ← H([v_1]_M, ..., [v_k]_M, m)$
9: $c ← Enc(c')$
10: $z ← y + sc$
11: if $z \notin \mathcal{R}_{q,[B-L_E]}$ then
12:   counter++
13:   Restart at step 3
14: end if
15: for $i = 1, ..., k$ do
16:   $w_i ← v_i - e_i c \mod q$
17:   if $\|w_i\|_\infty > 2^d - L_E \lor \|w_i\|_\infty > \lfloor q/2 \rfloor - L_E$ then
18:     counter++
19:     Restart at step 3
20: end if
21: end for
22: return $(c', z)$
qTESLA – Signing

Require: message $m$, and secret key $sk = (s, e_1, ..., e_k, seed_a, seed_y)$
Ensure: signature $(c', z)$

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

Require: message $m$, and secret key $sk = (s, e_1, ..., e_k, seed_u, seed_y)$
Ensure: signature $(c', z)$

1: counter ← 0
2: rand ← PRF₁(seed_y, $m$)
3: $y ← PRF₂(rand, counter)$
4: $a_1, ..., a_k ← GenL(seed_u)$
5: for $i = 1, ..., k$ do
6: \[ v_i = a_i y \mod q \]
7: end for
8: $c' ← H([v_1]_M, ..., [v_k]_M, m)$
9: $c ← Enc(c')$
10: $z ← u + sc$
11: if $z ∉ \mathbb{R}_{q, [B-L_E]}$ then
12: \[ \text{counter}++ \]
13: \[ \text{Restart at step 3} \]
14: end if
15: for $i = 1, ..., k$ do
16: \[ w_i ← v_i - e_i c \mod q \]
17: if $\|w_i\|_L > 2^d - L_E \lor \|w_i\|_∞ > \lfloor q/2 \rfloor - L_E$ then
18: \[ \text{counter}++ \]
19: \[ \text{Restart at step 3} \]
20: end if
21: end for
22: return $(c', z)$

Pseudo-randomness expansion
Computing sparse polynomial $c$ and candidate signature $z$
qTESLA – Signing

Require: message $m$, and secret key $sk = (s, e_1, ..., e_k, seed_u, seed_v)$
Ensure: signature $(c', z)$

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

Pseudo-randomness expansion

Computing sparse polynomial $c$ and candidate signature $z$

“security check“ = rejection sampling
qTESLA – Signing

Require: message $m$, and secret key $sk = (s, e_1, ..., e_k, seed_u, seed_y)$
Ensure: signature $(c', z)$

1: counter ← 0
2: rand ← PRF₁(seed_y, $m$)
3: $y ← PRF₂$(rand, counter)
4: $a_1, ..., a_k ← GenA$(seed_u)

5: for $i = 1, ..., k$ do
6: \[ v_i = a_iy \mod q \]
7: end for
8: \[ c' ← H([v_1]_M, ..., [v_k]_M, m) \]
9: $c ← Enc(c')$
10: $z ← u + sc$

11: if $z ∉ \mathcal{R}_{q,[N-L_E]}$ then
12: \[ counter++ \]
13: Restart at step 3
14: end if

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

22: return $(c', z)$

---

Pseudo-randomness expansion

Computing sparse polynomial $c$ and candidate signature $z$

“security check“ = rejection sampling

“correctness check“
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

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>Approach</th>
<th>Public key</th>
<th>Secret key</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>qTESLA-128</td>
<td>Heuristic</td>
<td>2 976</td>
<td>1 856</td>
<td>2 720</td>
</tr>
<tr>
<td>qTESLA-p-128</td>
<td>Provable</td>
<td>19 872</td>
<td>7 744</td>
<td>3 104</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>Approach</th>
<th>keygen</th>
<th>sign</th>
<th>verify</th>
</tr>
</thead>
<tbody>
<tr>
<td>qTESLA-128</td>
<td>Heuristic</td>
<td>2 234</td>
<td>1 279</td>
<td>229</td>
</tr>
<tr>
<td>qTESLA-p-128</td>
<td>Provable</td>
<td>4 334</td>
<td>2 420</td>
<td>732</td>
</tr>
</tbody>
</table>
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!