qTesla

Carl Miller
NIST Computer Security Division
June 1, 2018

NIST PQC Seminar (not for public distribution)

The Basics

- It’s a digital signature scheme.

- It is an LWE scheme (Learning With Errors) in the ring

$$\mathcal{R}_q := \mathbb{Z}_q[x]/\left(x^n + 1\right)$$

- It involves a trick using a hash function (forcing the signer to do steps in a certain order?).

Building Blocks

The Ring *R_q*

Let n be a power of two (e.g., 2048) and let q be a positive integer (e.g., 12,681,217). Consider the ring

$$\mathcal{R}_q := \mathbb{Z}_q[x]/\left(x^n + 1\right)$$

Let us say that a vector $v = v_0 + v_1 x + v_2 x^2 + \ldots$ in *R_q* is **short** if every v_i has small absolute value.
Let us say that such a vector v is **small** if $|v_i| <= 1$ for all i, *and* most of the values v_i are zero.

The Ring R_q

Suppose that we secretly pick a random short vector s in R_q.
We then publicly pick random elements a_1, a_2, a_3, … and compute

$$sa_1,\ sa_2,\ sa_3,\ \ldots$$

If we reveal these values to an adversary, she can easily determine s.
But if we mask them each with random small vectors

$$sa_1+e_1,\ sa_2+e_2,\ sa_3+e_3,\ \ldots$$

then determining s becomes a lot harder.

The Ring R_q

Basic hardness assumption: The distribution of

$(a, sa + e)$

is computationally indistinguishable from random.

Different forms of the "hard" problem

Suppose we are given a vectors u,a in R_q and are asked to find a short vector z such that

$$u \approx az$$

(That is, (u – az) is a short vector.)

This must be hard too (otherwise the problem on the previous page could be easily solved).

Different forms of the "hard" problem

Suppose we are given a vectors u,a in R_q and are asked to find a short vector z such that

$$u \approx az$$

(That is, (u – az) is a short vector.)

Next suppose that we are given a and allowed to pick u, <u>but</u> it must be of the form

$$u := w - H([w]_M)$$

where H is a hash function and $[\ ]_M$ = "most significant bits."

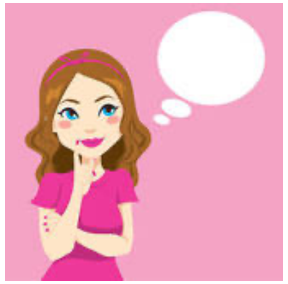
"Forging" for the upcoming sig.-prot. is similar to solving this.

Protocol

Overview

The signer produces two random-looking elements a,t in R_q (except that a is invertible).

There is a fixed hash function H that maps bit-strings to **small** elements of R_q.



Signer
Public key: a,t

Verifier

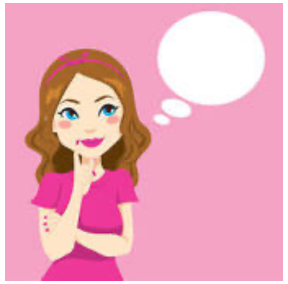
Public key: a,t

Overview

The signer signs a message m with a signature (z,c) where z, c are in R_q, z is short and c is small.

The verifier computes $w := az - tc$, and accepts only if

$$c = H([w]_M, m).$$



Signer
Public key: a,t

Message: m
Signature: c,z

Verifier

Public key: a,t

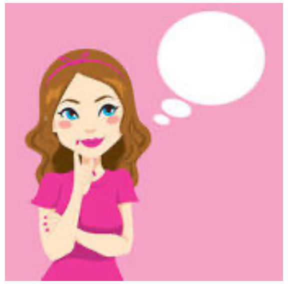
Overview

"Faking" a solution to the system

$$w = az - tc \qquad \text{and} \qquad c = H([w]_M, m)$$

is hard (?).

But given specific knowledge about how a,t were generated (specifically, if $t = as + e$, where s and e are short) it's easy.



Message: m
Signature: c,z

Signer
Public key: a,t
Secret key: s,e

Verifier
Public key: a,t

Procedures

Algorithm 1 Informal description of the key generation

Require: -

Ensure: Secret key $sk = (s, e, a)$, public key $pk = (a, t)$

1: $a \leftarrow \mathcal{R}_q$ invertible ring element
2: Choose $s, e \in \mathcal{R}$ with entries from $\mathcal{D}_\sigma$.
3: If the h largest entries of e sum to L_E then sample new e and retry at step 2.
4: If the h largest entries of s sum to L_S then sample new s and retry at step 2.
5: $t = as + e \in \mathcal{R}_q$.
6: Return secret key $sk = (s, e)$ and public key $pk = (a, t)$.

Gaussian distribution

Definition of t.

Procedures

Algorithm 2 Informal description of the signature generation

Require: Message m, secret key $sk = (s, e, a)$,
Ensure: Signature (z, c).

1: Choose y uniformly at random among B-short polynomials in $\mathcal{R}_q$.
2: $c \leftarrow H([ay]_M, m)$.
3: $z \leftarrow y + sc$.
4: If z is not $(B - L_S)$-short then retry at step 1.
5: If $ay - ec$ is not well-rounded then retry at step 1.
6: Return signature (z, c).

A solution is constructed to the system from 2 slides ago.
If y, s are short and e is small, then y+sc is short, as desired.

Procedures

Algorithm 2 Informal description of the signature generation

Require: Message m, secret key $sk = (s, e, a)$,
Ensure: Signature (z, c).

1: Choose y uniformly at random among B-short polynomials in $\mathcal{R}_q$.
2: $c \leftarrow H([ay]_M, m)$.
3: $z \leftarrow y + sc$.
4: If z is not $(B - L_S)$-short then retry at step 1.
5: If $ay - ec$ is not well-rounded then retry at step 1.
6: Return signature (z, c).

Randomness is needed here. In the full protocol, this randomness is drawn by hashing the message itself.

Performance

Security claims

The authors claim that their protocol is provably secure in the Quantum Random Oracle Model (QROM). This is established mostly by referring to other papers.

E. Alkim, et al. "Revisiting TESLA in the quantum random oracle model." https://eprint.iacr.org/2015/755.pdf (2017)

The protocol is in a sense a Fiat-Shamir transformation of a certain identification scheme (?). This is another way to approach security (?).

Security claims

The security proofs are based on a few assumptions, including the hardness of their version of Ring-LWE.

(Question: What hardness assumptions are made about the hash function?)

Numerical claims about security are based on the “LWE-Estimator” software.

Authors’ Response to Comment

- 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

Speed

Scheme	keygen	sign	verify	total (sign + verify)
qTESLA-128	3 402	2 495	520	3 015
qTESLA-192	5 875	9 686	1 065	10 751
qTESLA-256	12 433	26 063	1 310	38 496

Table 3: Performance (in thousands of cycles) of qTESLA on a 2.40 GHz Intel Core i5-6300U (Skylake) processor. Cycle counts are rounded to the nearest 10^3 cycles.

(These schemes address security levels 1, 3, and 5, respectively.)

Size

Some variables (such as “a”) are not stored as-is – a shorter bit string is stored and then and expanded using cSHAKE.

Table 2: Different key and signature sizes of our proposed parameter sets; we abbreviate theoretical sizes with TS and sizes as used in the implementations with IS; sizes are given in bytes.

Parameter set	TS/IS	public key	secret key	signature
qTesla-128	TS	2 976	1 856	2 720
	IS	4 128	2 112	3 104
qTesla-192	TS	6 176	4 160	5 664
	IS	8 224	8 256	6 176
qTesla-256	TS	6 432	4 128	5 920
	IS	8 224	8 256	6 176

The authors claim to have one of the smallest signature sizes against a quantum adversary.

qTesla

Carl Miller
NIST Computer Security Division
June 3, 2018

NIST PQC Seminar (not for public distribution)