Implementation of Lattice-Based Signature Scheme Ring-Tesla and Comparison with ECSDA

Kenneth Xu
Stanford University

Abstract

In search of efficient post-quantum alternative signature schemes, lattice-based schemes like BLISS and GLP have become promising fields of research. In this paper we provide an open-source implementation of the Ring-TESLA scheme [1], which is based on the TESLA signature scheme by Alkim et al. [2]. Ring-TESLA is not only theoretically as efficient as the BLISS and GLIP schemes, but also has provably secure instantiation.
We compare the speed of our Ring-Tesla implementation with an ECDSA implementation for two different security levels.

1 Introduction

Our implementation closely follows the description provided by Akleylet et al. 2016. Ring-TESLA has a security reduction from the R-LWE problem [3] [1]. As long as R-LWE is computationally hard, Ring-Tesla is unforgeable against the chosen-message attack. [1]

1.1 Advantages over BLISS and GLP

Ring-Tesla has a stronger security argument since it achieves both good performance with provably secure instantiation, while BLISS and GLP can only achieve one or the other. Provably secure instantiation means parameters are chosen according to the security reduction [1].
Moreover, Ring-Tesla uses uniform-sampling during signature generation, unlike BLISS, which uses Gaussian-sampling, generally assumed to be vulnerable to timing attacks. Comparing the resilience of BLISS, GLP, and Ring-TESLA to fault attacks in Bindel et al. 2016 [4] found that the Ring-TESLA scheme was sensitive to a strict subset of fault attacks affecting the BLISS and GLP.

2 Ring-Tesla Signature Scheme

Ring-Tesla is parameterized by a number of integers: $n, \omega, d, B, q, U, L, \kappa$ and the security parameter λ where $n > \kappa > \lambda$. n is a positive power of 2 and q is a prime where $q = 1(\mod 2n)$.
The quotient ring of polynomials we work with is defined as $R_q = \mathbb{Z}_q[x]/(x^n+1)$ - in other words, all polynomials of degree up to $n-1$ with coefficients in the range $(-\frac{q}{2}, \frac{q}{2})$. The signature scheme also uses a Gaussian distribution D_σ (with standard deviation σ), a Hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\kappa$, and an encoding function F which maps the binary output of H to a vector of length n and weight ω. We implemented a similar encoding scheme as found in Gneysu et al. [5]

We provide a mathematical overview of the Ring-Tesla algorithm below:

2.1 Globals

a_1 and a_2 are global polynomials uniformly sampled from R_q.

2.2 Key Generation

The pseudocode below is borrowed from Akleylet et al. [1]

```
KeyGen(1^λ;a_1,a_2):
  s,e_1,e_2 → D_σ^n
  If checkE(e_1) = 0 ∨ checkE(e_2) = 0:
    Restart
  t_1 → a_1 s + e_1 (mod q)
  t_2 → a_2 s + e_2 (mod q)
  sk → (s,e_1,e_2)
  pk → (t_1,t_2)
  return (sk, pk)
```

We first sample three polynomials s, e_1, and e_2 from Gaussian distribution D_σ. Each polynomial requires n

samples, one for each degree from 0 to $n-1$.
A polynomial passes the checkE function if the sum of its ω largest coefficients is less than L.

2.3 Sign

The pseudocode below is borrowed from Akleylet et al. [1]

Sign(μ;a_1,a_2,s,e_1,e_2):
  y $\rightarrow$ $R_{q,[B]}$
  v_1 $\rightarrow$ a_1y (mod q)
  v_2 $\rightarrow$ a_2y (mod q)
  c' $\rightarrow$ H$\left(\lfloor v_1 \rceil_{d,q},\ \lfloor v_2 \rceil_{d,q}\right)$
  c $\rightarrow$ F(c')
  z $\rightarrow$ y + sc

  # Rejection sampling
  w_1 $\rightarrow$ v_1 - e_1c (mod q)
  w_2 $\rightarrow$ v_2 - e_2c (mod q)
  If $[w_1]_{2^d}$,$[w_2]_{2^d} \notin R_{2^d-L}$ $\vee$ $z \notin R_{B-U}$:
    Restart
  Return (z, c')

First, polynomial y is uniformly sampled from R_q, with additional constraints on the size of coefficients. Every coefficient in y must lie in the range $[-B,B]$ where $B \in [0, \frac{q}{2}]$.
We hash the concatenation of the rounded values of v_1 and v_2 and the message μ (to sign). This rounding function is defined the following way: $\lfloor x \rceil_{d,q} = \lfloor x(\bmod q) \rceil_d$ and $\lfloor y \rceil_d = (y - [y]_{2^d})/2^d$, where $[y]_{2^d}$ is the mod representation of y in the range $[-2^{d-1}, 2^{d-1}]$ and $/2^d$ defines a quotient group.
The encoding function is applied right after hashing to produce the signature (z,c).
Before returning, however, we apply rejection sampling by making sure the coefficients of polynomials w_1, w_2, and z are not too large.

2.4 Verify

The pseudocode below is borrowed from Akleylet et al. [1] and is similar to the reverse of sign().

Verify(μ;z,c';a_1,a_2,t_1,t_2):
  c $\rightarrow$ F(c')
  w'_1 $\rightarrow$ a_1z - t_1c (mod q)
  w'_2 $\rightarrow$ a_2z - t_2c (mod q)
  c'' $\rightarrow$ H$\left(\lfloor w'_1 \rceil_{d,q},\ \lfloor w'_2 \rceil_{d,q}\right)$
  If c'=c'' $\wedge$ z $\in R_{B-U}$:
    Return 1
  Else: Return 0

3 Implementation

We implemented the Ring-Tesla signature scheme in C++ and compared its speed with that of a ECDSA C++ implementation.

3.1 Selection of Parameters

We selected mostly the same provably secure parameters as Akleylet et al. [1] described, for two security levels: 80-bit and 128-bit. Like the paper, we will name them RingTesla-I and RingTesla-II respectively. We changed the value of ω to 16 for easier implementation. Below are our selected parameters:

Parameter set	Security bits	n	σ	L	ω	B	U	d	q
RingTesla-I	80	512	30	814	16	$2^{21}-1$	993	21	8399873
RingTesla-II	128	512	52	2766	16	$2^{22}-1$	3173	23	39960577

3.2 Code

Our implementation is available at:
https://github.com/kenxu95/rtesla

The code also contains speed an soundness tests (in main.cc).

4 Performance Results

We ran and compared the speed of the three methods KeyGen(), Sign(), and Verify() on 10,000 random string messages of 500 characters. We measured the speed in cpu time. To calculate the speed of each call, we averaged over all 10,000 trials for each method. Below are the results (in milliseconds per call):

Signature Scheme	KeyGen	Sign	Verify
Ring-Tesla-I	2.61	16.20	4.99
Ring-Tesla-II	2.49	15.30	4.82
ECDSA w/ secp160r1	0.92	1.03	1.08
ECDSA w/ secp192r1	0.67	1.07	1.15
ECDSA w/ secp224r1	1.71	1.89	2.05
ECDSA w/ secp256r1	2.81	2.99	3.29
ECDSA w/ secp256k1	2.14	2.30	2.35

We compared our Ring-Tesla implementation against five different ECDSA curves, each providing a different level of bit security. While the trend is for the ECDSA algorithm to take more time when providing more bits of security, the same cannot be said for Ring-Tesla.

We are mainly interested in curves secp192r1 and secp256r1, which provide 80 and 128 bits of security respectively, just like our Ring-Tesla parameter sets.
It is clear that Ring-Tesla is significantly slower when signing, possibly due to rejection sampling. For 80-bit

security, key generation and verification in ECDSA are a few times faster. However, for 128-bit security, the gap between the two schemes shrinks significantly. It is possible that, with sufficient optimization, Ring-Tesla can match the speed of ECSDA key generation and verification.

References

[1] AKLEYLEK, S., BINDEL, N., BUCHMANN, J., KRMER, J., AND MARSON, G. A. An efficient lattice-based signature scheme with provably secure instantiation. In *International Conference on Cryptology - AFRICACRYPT 2016* (2016), A. N. D. Pointcheval, T. Rachidi, Ed., Springer, pp. 44–60.

[2] ALKIM, E., BINDEL, N., BUCHMANN, J., AND DAGDELEN, O. Tesla: Tightly-secure efficient signatures from standard lattices. In *Cryptology ePrint Archive* (2015).

[3] ALKIM, E., BINDEL, N., BUCHMANN, J., AND ÖZGÜR DAGDELEN. Tesla: Tightly-secure efficient signatures from standard lattices.

[4] BINDEL, N., BUCHMANN, J. A., AND KRMER, J. Lattice-based signature schemes and their sensitivity to fault attacks. In *IACR Cryptology ePrint Archive* (2016).

[5] GNEYSU, T., LYUBASHEVSKY, V., AND PPPELMANN, T. Practical lattice-based cryptography: A signature scheme for embedded systems. In *Cryptographic Hardware and Embedded Systems – CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings* (Berlin, Heidelberg, 2012), E. Prouff and P. Schaumont, Eds., Springer Berlin Heidelberg, pp. 530–547.

Notes

[1]Gus Gutoski and Chris Peikert discovered a mistake in the tight security reduction from the R-LWE problem to TESLA presented in the referenced paper. The mistake, however, does not yet lead to any attack against TESLA. The non-tight security reduction given by Bai and Galbraith still holds.