CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme

Léo Ducas (CWI), Eike Kiltz (Ruhr-Universität Bochum),
Tancrède Lepoint (SRI International), Vadim Lyubashevsky (IBM Research),
Peter Schwabe (Radboud University), **Gregor Seiler (IBM Research)**,
Damien Stehlé (ENS de Lyon)

September 10, 2018

Overview

- Signature scheme submitted to the NIST PQC standardization process

Overview

- Signature scheme submitted to the NIST PQC standardization process
  - One out of 5 lattice-based signature schemes

Overview

- Signature scheme submitted to the NIST PQC standardization process
  - One out of 5 lattice-based signature schemes
- Public key size 1.5 KB, signature size 2.7 KB (recommended parameters)

Overview

- Signature scheme submitted to the NIST PQC standardization process
  - One out of 5 lattice-based signature schemes
- Public key size 1.5 KB, signature size 2.7 KB (recommended parameters)
- Design based on "Fiat-Shamir with Aborts" technique [Lyu09]

Overview

- Signature scheme submitted to the NIST PQC standardization process
  - One out of 5 lattice-based signature schemes
- Public key size 1.5 KB, signature size 2.7 KB (recommended parameters)
- Design based on "Fiat-Shamir with Aborts" technique [Lyu09]
  - Rejection sampling is used to sample signatures that do not reveal secret information

Overview

- Signature scheme submitted to the NIST PQC standardization process
  - One out of 5 lattice-based signature schemes
- Public key size 1.5 KB, signature size 2.7 KB (recommended parameters)
- Design based on “Fiat-Shamir with Aborts” technique [Lyu09]
  - Rejection sampling is used to sample signatures that do not reveal secret information
- Signature compression as developped in [GLP12], [BG14] ($>$ 50% smaller)

Overview

- Signature scheme submitted to the NIST PQC standardization process
  - One out of 5 lattice-based signature schemes
- Public key size 1.5 KB, signature size 2.7 KB (recommended parameters)
- Design based on "Fiat-Shamir with Aborts" technique [Lyu09]
  - Rejection sampling is used to sample signatures that do not reveal secret information
- Signature compression as developped in [GLP12], [BG14] ($>$ 50% smaller)
- *New:* Compression of public key (60% smaller, 100 byte larger signature)

Overview

- Signature scheme submitted to the NIST PQC standardization process
  - One out of 5 lattice-based signature schemes
- Public key size 1.5 KB, signature size 2.7 KB (recommended parameters)
- Design based on “Fiat-Shamir with Aborts” technique [Lyu09]
  - Rejection sampling is used to sample signatures that do not reveal secret information
- Signature compression as developped in [GLP12], [BG14] ($>$ 50% smaller)
- *New:* Compression of public key (60% smaller, 100 byte larger signature)
- *New:* Hardness based on *Module*-LWE/SIS

Overview

- Signature scheme submitted to the NIST PQC standardization process
  - One out of 5 lattice-based signature schemes
- Public key size 1.5 KB, signature size 2.7 KB (recommended parameters)
- Design based on “Fiat-Shamir with Aborts” technique [Lyu09]
  - Rejection sampling is used to sample signatures that do not reveal secret information
- Signature compression as developped in [GLP12], [BG14] ($>$ 50% smaller)
- *New:* Compression of public key (60% smaller, 100 byte larger signature)
- *New:* Hardness based on *Module*-LWE/SIS
- *New:* Very efficient implementation

Principal Design Considerations

- Easy to implement securely – No Gaussian sampling
- Small total size of public key + signature
  - Among the smallest total size of all NIST submissions (Falcon is smaller)
- Conservative parameter selection
- Modular design
  - Use of Module-LWE/SIS allows to work over the same small ring for all security levels: Arithmetic needs only be optimized once and for all

Choice of Ring

Strategy: Choose smallest ring dimension n that gives main advantages of Ring-LWE

Choice of Ring

Strategy: Choose smallest ring dimension n that gives main advantages of Ring-LWE

Dimension $n = 256$ is enough to get sufficiently large set of small norm challenges

Fully splitting prime q allows for NTT-based multiplication (more about this later)

$$\boxed{R = \mathbb{Z}_{2^{23}-2^{13}+1}[X]/(X^{256}+1)}$$

Simplified Scheme

Key generation:

$$\mathbf{A} \leftarrow R^{5\times 4}$$

$$\mathbf{s}_1 \leftarrow S_5^4,\ \ \mathbf{s}_2 \leftarrow S_5^5$$

$$\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$$

$$pk = (\mathbf{A}, \mathbf{t}),\ \ sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2)$$

Verification:

$$c' = \mathrm{H}(\mathrm{High}(\overbrace{\mathbf{A}\mathbf{z} - c\mathbf{t}}^{=\mathbf{w}-c\mathbf{s}_2}), M)$$

If $\|\mathbf{z}\|_\infty \leq \gamma - \beta$ and $c' = c$, accept

Signing:

$$\mathbf{y} \leftarrow S_\gamma^4$$

$$\mathbf{w} = \mathbf{A}\mathbf{y}$$

$$c = \mathrm{H}(\mathrm{High}(\mathbf{w}), M) \in B_{60}$$

$$\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$$

If $\|\mathbf{z}\|_\infty > \gamma - \beta$ or $\|\mathrm{Low}(\mathbf{w} - c\mathbf{s}_2)\|_\infty > \gamma - \beta$, restart

$$sig = (\mathbf{z}, c)$$

Public Key Compression

Verification:

$c' = \mathrm{H}(\mathrm{High}(\mathbf{Az} - c\mathbf{t}), M)$

If $\|\mathbf{z}\|_\infty \leq \gamma - \beta$ and $c' = c$, accept

Decompose $\mathbf{t} = \mathbf{t}_1 2^{14} + \mathbf{t}_0$ and put only $\mathbf{t}_1$ into public key ($23 \rightarrow 9$ bits per coefficient)

Public Key Compression

Verification:

$c' = \mathrm{H}(\mathrm{High}(\mathbf{Az} - c\mathbf{t}), M)$

If $\|\mathbf{z}\|_\infty \leq \gamma - \beta$ and $c' = c$, accept

Decompose $\mathbf{t} = \mathbf{t}_1 2^{14} + \mathbf{t}_0$ and put only $\mathbf{t}_1$ into public key ($23 \rightarrow 9$ bits per coefficient)

For verification we need to compute

$$\mathrm{High}(\mathbf{Az} - c\mathbf{t}) = \mathrm{High}(\mathbf{Az} - c\mathbf{t}_1 2^{14} - c\mathbf{t}_0)$$

Include carries from adding $-c\mathbf{t}_0$ in signature $\rightarrow$ $\mathrm{High}(\mathbf{Az} - c\mathbf{t}_1 2^{14})$ can be corrected

Security

Tight reduction, even in quantum random oracle model, from *SelfTargetMSIS* and Module-LWE/SIS [KLS18]:

$$\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}(A) \leq \mathsf{Adv}^{\mathsf{MLWE}}(B) + \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}(C) + \mathsf{Adv}^{\mathsf{MSIS}}(D) + 2^{-254}$$

Given matrix $\mathbf{A}$, find short vector $\mathbf{y}$, challenge polynomial c and message M such that

$$\mathsf{H}\left((\mathbf{I} \mid \mathbf{A})\begin{pmatrix}\mathbf{y}\\ c\end{pmatrix}, M\right) = c$$

SelfTargetMSIS has non-tight reduction with standard forking lemma argument from Module-SIS

Implementation

Reference and AVX2 optimized implementations on

`https://github.com/pq-crystals/dilithium`

Main Operations:

- Polynomial multiplication in fixed ring $R = \mathbb{Z}_{2^{23}-2^{13}+1}[X](X^{256}+1)$
- Expansion of the SHAKE XOF
  - Independent sampling of polynomials: Allows for parallel use of SHAKE

Constant Time

Our implementations are fully protected against timing side channel attacks

In particular: No use of the C '%'-operator

Note: Sampling of challenge polynomials is not constant-time and does not need to be

Speed of Reference Implementation

	Key generation	Signing	Signing (average)	Verification
Multiplication	89, 591	987, 666	1, 280, 053	143, 924
SHAKE	178, 487	314, 570	377, 068	161, 079
Modular Reduction	11, 944	120, 793	163, 017	10, 626
Rounding	6, 586	108, 412	137, 324	11, 821
Rejection Sampling	60, 740	76, 893	94, 607	28, 082
Addition	8, 008	58, 696	79, 498	10, 723
Packing	7, 114	17, 183	18, 856	8, 883
Total	381, 178	1, 778, 148	2, 260, 429	396, 043

Median cycles of 5000 executions on Intel Skylake i7-6600U processor

Advantages of NTT Multiplication

NTT-based multiplication allows for easy reuse of computation:

- In Dilithium on average about 224 multiplications to sign a message

Advantages of NTT Multiplication

NTT-based multiplication allows for easy reuse of computation:

- In Dilithium on average about 224 multiplications to sign a message
- So, naively, 673 NTTs

Advantages of NTT Multiplication

NTT-based multiplication allows for easy reuse of computation:

- In Dilithium on average about 224 multiplications to sign a message
- So, naively, 673 NTTs
- But we only actually perform 172 NTTs

Advantages of NTT Multiplication

NTT-based multiplication allows for easy reuse of computation:

- In Dilithium on average about 224 multiplications to sign a message
- So, naively, 673 NTTs
- But we only actually perform 172 NTTs

Advantages of NTT Multiplication

NTT-based multiplication allows for easy reuse of computation:

- In Dilithium on average about 224 multiplications to sign a message
- So, naively, 673 NTTs
- But we only actually perform 172 NTTs

We immediately get a 4x speed-up in multiplication time from saving NTTs compared to Karatsuba multiplication

Note: In our reference implementation NTTs still make up for the most time comsuming operation

AVX2 optimized Implementation

Optimizations:

- Vectorized NTT in assembly
- 4-way parallel SHAKE
- Better public key and signature compression
- Faster assembly modular reduction

AVX2 optimized Implementation

Optimizations:

- Vectorized NTT in assembly
- 4-way parallel SHAKE
- Better public key and signature compression
- Faster assembly modular reduction

About 3.5x faster signing compared to reference version

AVX2 optimized Implementation

Optimizations:

- Vectorized NTT in assembly
- 4-way parallel SHAKE
- Better public key and signature compression
- Faster assembly modular reduction

About 3.5x faster signing compared to reference version

Recent update: $>$ 40% faster compared to TCHES paper

New Fast Vectorized NTT Implementation

Prior state of the art: Double floating point arithmetic as in NewHope

Now: Fast approach with integer arithmetic and same Montgomery reduction strategy as in reference implementation

New Fast Vectorized NTT Implementation

Prior state of the art: Double floating point arithmetic as in NewHope

Now: Fast approach with integer arithmetic and same Montgomery reduction strategy as in reference implementation

Unfortunately not as fast as 16-bit NTT in Kyber because of missing instruction for high product

New Fast Vectorized NTT Implementation

Prior state of the art: Double floating point arithmetic as in NewHope

Now: Fast approach with integer arithmetic and same Montgomery reduction strategy as in reference implementation

Unfortunately not as fast as 16-bit NTT in Kyber because of missing instruction for high product

	Dilithium	Floating point	Kyber (16bit)	Saber (16bit)
NTT	$1,382$	$2,989$	393	—
Inverse NTT	$1,292$	$3,215$	366	—
Full multiplication	$4,288$	$10,042$	$1,162$	$3,810$

Roughly 2x speed-up over floating point NTT

Speed of AVX2 optimized Implementation

	Key generation	Signing	Signing (average)	Verification
Multiplication	15, 794	155, 721	201, 347	25, 471
SHAKE	96, 779	170, 232	205, 847	90, 921
Modular reduction	1, 034	7, 902	10, 541	708
Rounding	728	7, 541	9, 904	2, 479
Rejection sampling	62, 272	67, 193	81, 278	27, 737
Addition	8, 028	46, 755	62, 453	8, 659
Packing	6, 997	16, 200	17, 526	8, 712
Total	199, 306	510, 298	635, 019	174, 951

Questions?

Module LWE (aka Generalized LWE)

Polynomial ring: $R = \mathbb{Z}_q[X]/(X^n + 1)$

It is hard to distinguish between uniform vector $\mathbf{t} \in R^k$ and $\mathbf{t}$ of the form

$$\mathbf{t} = \begin{pmatrix} t_1 \\ \vdots \\ t_k \end{pmatrix} = \underbrace{\begin{pmatrix} a_{1,1} & \dots & a_{1,l} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \dots & a_{k,l} \end{pmatrix}}_{\text{uniform, public}} \underbrace{\begin{pmatrix} s_{1,1} \\ \vdots \\ s_{1,l} \end{pmatrix}}_{\text{short}} + \underbrace{\begin{pmatrix} s_{2,1} \\ \vdots \\ s_{2,k} \end{pmatrix}}_{\text{short}}$$

Conservative parameters: Coefficients of $s_{i,j}$ are from $\{-5, \dots, 5\}$

- $\mathbf{s}_1$ lives in a *module* over R of rank l
- Ring-LWE is special case where $l = 1$ and $\mathbf{s}_1$ lies in the *ring* R
- Plain LWE is special case when the dimension n of the ring is 1 so that $R = \mathbb{Z}_q$.
- Security: Effective dimension over $\mathbb{Z}_q$ is $l \cdot n$

NTT Multiplication

Suppose $\zeta \in \mathbb{Z}_q$ is a primitive 8-th root of unity, i.e. $\zeta^4 = -1$.

$$\mathbb{Z}_q[X]/(X^{256}+1)$$

$$\mathbb{Z}_q[X]/(X^{128}-\zeta^2) \qquad \mathbb{Z}_q[X]/(X^{128}+\zeta^2)$$

$$\mathbb{Z}_q[X]/(X^{64}-\zeta) \qquad \mathbb{Z}_q[X]/(X^{64}+\zeta) \qquad \mathbb{Z}_q[X]/(X^{64}-\zeta^3) \qquad \mathbb{Z}_q[X]/(X^{64}+\zeta^3)$$

Advantages of NTT Multiplication

Consider the matrix-vector product

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

This needs 20 multiplications or 60 NTTs for full NTT-based multiplications

With NTT-based multiplication, the $a_{i,j}$ can be directly sampled in their NTT representation

Also only one inverse NTT per row necessary

We only need to compute 9 NTTs for the matrix-vector product