Titanium: Post-Quantum Lattice-Based Public-Key Encryption balancing Security Risk and Practicality

Ron Steinfeld, Amin Sakzad, Raymond K. Zhao

Monash University

ron.steinfeld@monash.edu

Overview

Intro

Approaches to Lattice-based cryptography

Low Security Risk Oriented: LWE Approach:

- **Advantage**: Low Security Risk – no lattice structure, relation to worst-case lattices
- **Drawback**: Performance – large $\geq n \times n$ matrices, slow computation

High Performance Oriented: PLWE^f Approach

Problem (Search Poly. Learning-with-Errors *Search* – $PLWE^f_{q,m,n,\alpha}$)

Let $R_q = \mathbb{Z}_q[x]/(f(x))$ *(e.g.* $f(x) = x^n + 1$*). Given* $\mathbf{A} \hookleftarrow U(R_q^{m \times 1})$ *and* $\mathbf{y} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod R_q$ *(with* $\mathbf{e}$ *'small'), find* $\mathbf{s}$*.*

- **Advantage**: Performance – succinct matrices, fast poly arithmetic (FFT)
- **Drawback**: High Security Risk? – rely on security of PLWE^f for a **fixed** f...

Risk: PLWE^f/ApproxSVP^f easy for some f

Problem

ApproxSVP^f ***Problem***: *ApproxSVP restricted to ideals in* $R = \mathbb{Z}[x]/f(x)$

Weak f's for ApproxSVP^f:

- **The case of cyclotomic f of prime power index**:
  - [CDPR16]: quantum poly. time algorithm to find a **short** generator of a principal ideal for $2^{O(\sqrt{n})}$ approx. factor
  - [CDW17]: quantum poly. time algorithm to solve ApproxSVP for all ideals for $2^{O(\sqrt{n})}$ approx. factor
- **The case of multiquadratic f:**
  - [BBdVLvV17]: quasipoly. time algorithm to find a **short** generator of a principal ideal

Weak f's for PLWE^f **with 'small' noise**:

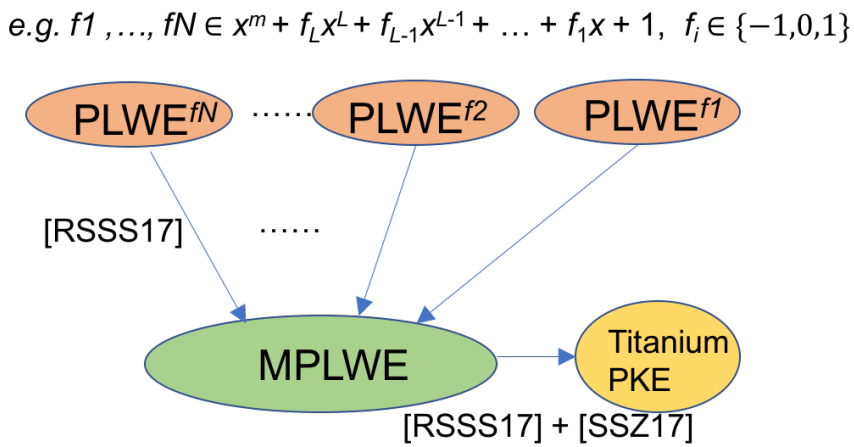
- [EHL14,ELOS15,CIV16,Pei16]: poly-time attacks on PLWE^f for weak f, when noise is 'small' (vs. canonical embeding lattice geometry)

Risk of fixing f today: future attacks on PLWE^f for weak f's? Which f?

Titanium: Our MP-LWE-based Approach (Risk-Performance Balance)

Risk-Performance Balance: Titanium MP-LWE-based Approach

- **Middle-Product LWE (MP-LWE) [RSSS17]:** poly. variant of LWE problem as secure as the **hardest** PLWE^f for **a big family** $\mathcal{F}$ of ring polynomials f's
- Lower Security Risk guarantee: hedge risk across class $\mathcal{F}$ of f's
- **Security-Risk-vs.-Perf. Balance:** Lower security risk guarantee than PLWE^f schemes, Better performance than LWE schemes



Titanium PKE/KEM (Security-Risk-Performance Balance)

- Our CPA-secure PKE scheme: **Titanium-CPA**
  - **Performance:** Optimised [RSSS17] PKE
    - Fast FFT-based algorithms for polynomial 'middle-product'
    - Optimised noise/randomness distributions/parameters
    - Constant-time implementation
  - **Security:** Optimised [RSSS17] sec. proof
    - MP-LWE-based: Low security risk from hardest PLWE^f over $f \in \mathcal{F}$
    - Used in parameter selection: concrete security lower bound guarantees
    - Conservative choice of parameters
- Our CCA-secure KEM scheme: **Titanium-CCA**
  - Tight CCA conversion (classical ROM) of Titanium-CPA using Fujisaki-Okamoto variant
  - Provable resistance to decryption failure attacks

Security Foundations: Middle-Product LWE (MP-LWE) Problem

Middle Product of two polynomials

Let R be a ring, $a \in R^{<n}[x]$ and $b \in R^{<n+d-1}[x]$ two polynomials.

- Their product is:

$$c_0 + \cdots + c_{n-2}x^{n-2}$$
$$+c_{n-1}x^{n-1}+c_n x^n + \cdots + c_{n+d-2}x^{n+d-2}$$
$$+c_{n+d-1}x^{n+d-1} + \cdots + c_{2n+d-3}x^{2n+d-3} \in R^{<2n+d-2}[x]$$

- Their **middle** product is:

$$a \odot_d b := c_{n-1} + c_n \cdot x + \cdots + c_{n+d-2} \cdot x^{d-1} \in R^{<d}[x]$$

PLWE and MP-LWE problems

Decision $\text{PLWE}^{f}_{q,\alpha,\chi}$ Problem

Let $s \hookleftarrow (U(\mathbb{Z}_q[x]/f))$, $a_i \hookleftarrow U(\mathbb{Z}_q[x]/f)$ and $e_i \hookleftarrow \chi^n$ 'small'.
Distinguish between $(a_i, b_i = a_i \cdot s + e_i)_i$ and $(a_i, b_i \hookleftarrow U(\mathbb{Z}_q[x]/f))_i$

Decision $\text{MP-LWE}^{n}_{q,\alpha,\chi,d}$ Problem

Let $s \hookleftarrow (U(\mathbb{Z}_q[x]^{<n+d-1}))$, $a_i \hookleftarrow U(\mathbb{Z}_q^{\leq n}[x])$ and $e_i \hookleftarrow \chi^d$ 'small'.
Distinguish between $(a_i, b_i = a \odot_d s + e)_i$ and $(a_i, b_i \hookleftarrow U(\mathbb{Z}_q^{\leq d}[x]))_i$

Hardness of MP-LWE

Let $n \geq 1$, $q \geq 2$, and $\alpha \in (0,1)$, χ balanced.

Theorem (Hardness of MP-LWE$^n_{q,\alpha,\chi,d'}$ (RSSS17 + SSZ17))

PLWE$^f_{q,\alpha,\chi}$ *reduces to* MP-LWE$^n_{q,\alpha,\chi,d'}$ *for* ***any*** *monic* $f \in \mathbb{Z}[x]$ *in family* $\mathcal{F}(n, m', d')$ *s.t.*

- $f(x) = x^m + \sum_{i \leq \ell(m)} f_i x^i$
- $\ell(m) = \min(m/2 + 1, m + 1 - d')$
- $d' \leq m' \leq m \leq n$
- $f_0 \in \{-1, 1\}$.

- Tight Reduction w.r.t. running-time, advantage, and preserves noise distribution
- Improves on noise amplifying reduction of [RSSS17]
- For Titanium, we use $\chi = \text{BinDiff}(\eta)$, diff. of binomials ($\approx$ Gaussian)

Polynomial Family $\mathcal{F}_1 = \mathcal{F}(n, m', d')$ of f for Titanium security foundation

$f(x) = x^m + \sum_{i \le \ell(m)} f_i x^i$,
Family degree range $m_{min} = m' \le m \le m_{max} = n$
Deg. of largest non-leading monomial $\ell(m') = \text{gap}_2 = m_{min} - d'$

Parameters of $\mathcal{F}_1$ for Titanium-CCA**:**

Parameter	Toy64	Lite96	Std128	Med160	Hi192	Super256
$m_{min} = m'$	654	770	896	1230	1486	1998
$m_{max} = n$	684	800	1024	1280	1536	2048
$\ell(m') = \text{gap}_2$	142	35	128	462	462	718
lo bnd on $\log_3(\lvert\mathcal{F}_1\rvert)$	172	65	256	512	512	768
power-of-two inclusion	×	×	✓	×	×	✓

Titanium Algorithms (Simplified Versions) and Par. Sets

Titanium-CPA Key Gen. Algorithm

Algorithm 1 : Titanium-CPA.KeyGen

Input: 1^λ.

Output: pk and sk.

1: **function** KeyGen(1^λ)
2: Let $s \hookleftarrow U(\mathbb{Z}_q^{<n+d+k-1}[x])$.
3: Let $(a_1, \ldots, a_t) \hookleftarrow U(\mathbb{Z}_q^{<n}[x])^t$.
4: Let $(e_1, \ldots, e_t) \hookleftarrow \chi_e \in (\mathbb{Z}_q^{<d+k}[x])^t$.
5: **for** $i \leq t$ **do**
6: Let $b_i = a_i \odot_{d+k} s + e_i \in \mathbb{Z}_q^{<d+k}[x]$.
7: **end for**
8: Let pk $= ((a_1, \ldots, a_t), (b_1, \ldots, b_t))$ and sk $= s$.
9: **end function**

Omitted from above version (using XOF = SHA-3 KMAC256 PRF/"RO"):

- Pseudorandom gen. of s and $(e_1, \ldots, e_t)$ from seedsk stored in sk
- "Pseudorandom" gen. of $(a_1, \ldots, a_t)$ from seedpk stored in pk
- a_i sampled in reversed coeff. format (for efficient MP algorithm)

Titanium-CPA Encryption. Algorithm

Algorithm 2 : Titanium-CPA.Encrypt

Input: $\mathsf{pk} = ((a_1, \ldots, a_t), (b_1, \ldots, b_t))$ and $\mathsf{m} \in \mathbb{Z}_p^{<d}[x]$.
Output: $\mathsf{ct} = (c_1', c_2')$.

1: **function** Encrypt(pk, m)
2: Let $(r_1, \ldots, r_t) \hookleftarrow \chi_r \in (\mathbb{Z}_q^{<k+1}[x])^t$.
3: Let $c_1' = \sum_{i=1}^{t} r_i \cdot a_i$
4: Let $c_2' = \sum_{i=1}^{t} r_i \odot_d b_i + \lfloor q/p \rfloor \cdot m \in \mathbb{Z}_q^{<d}[x]$.
5: **end function**

Omitted from above version (using XOF = SHA-3 KMAC256 PRF/RO):

- Pseudorandom generation of $(r_1, \ldots, r_t)$ from seedr
- "Pseudorandom" gen. of $(a_1, \ldots, a_t)$ from seedpk stored in pk
- a_i sampled in reversed coeff. format, $r_i \odot_d b_i$ replaced by $\mathrm{Rev}(r_i) \odot_d b_i$ (for efficient MP algorithm)

Titanium-CPA Decryption Algorithm

Algorithm 3 : Titanium-CPA.Decrypt

Input: sk $= s$ and ct $= (c'_1, c'_2)$.
Output: m'.

1: **function** Decrypt(sk, ct)
2: Let $c' = c'_2 - c'_1 \odot_d s \in \mathbb{Z}_q^{<d}[x]$.
3: Let $m' = \text{Round}(\lfloor q/p \rfloor, c') \in \mathbb{Z}_p^{<d}[x]$.
4: **end function**

Omitted from above version (using XOF = SHA-3 KMAC256 PRF/RO):

- Pseudorandom gen. of s from seedsk stored in sk
- $c'_1 \odot_d s$ replaced by $\text{Rev}(c'_1) \odot_d s$ (for efficient MP algorithm)

Titanium-CCA: Key Encapsulation Mechanism

- Generic Fujisaki-Okamoto CPA to CCA transformation applied to Titanium-CPA [HHK07]
  - Decryption returns pseudorandom value if ciphertext validity check fails
- Uses two hash functions G,H = SHA-3 based SHAKE256 "random oracles"

Correctness of Titanium-CPA

Looking at decryption Algorithm:

$$c' \quad = \quad c_2' - c_1' \odot_d s$$

Correctness of Titanium-CPA

Looking at decryption Algorithm:

$$\begin{aligned} c' &= c_2' - c_1' \odot_d s \\ &= \sum_{i=1}^{t} r_i \odot_d b_i + \lfloor q/p \rfloor \cdot m - \left(\sum_{i=1}^{t} r_i \cdot a_i \right) \odot_d s \end{aligned}$$

Correctness of Titanium-CPA

Looking at decryption Algorithm:

$$
\begin{aligned}
c' &= c_2' - c_1' \odot_d s \\
&= \sum_{i=1}^{t} r_i \odot_d b_i + \lfloor q/p \rfloor \cdot m - \left(\sum_{i=1}^{t} r_i \cdot a_i \right) \odot_d s \\
&= \sum_{i=1}^{t} r_i \odot_d (a_i \odot_{d+k} s + e_i) + \lfloor q/p \rfloor \cdot m - \sum_{i=1}^{t} (r_i \cdot a_i) \odot_d s
\end{aligned}
$$

Correctness of Titanium-CPA

Looking at decryption Algorithm:

$$\begin{aligned} c' &= c_2' - c_1' \odot_d s \\ &= \sum_{i=1}^{t} r_i \odot_d b_i + \lfloor q/p \rfloor \cdot m - \left(\sum_{i=1}^{t} r_i \cdot a_i \right) \odot_d s \\ &= \sum_{i=1}^{t} r_i \odot_d (a_i \odot_{d+k} s + e_i) + \lfloor q/p \rfloor \cdot m - \sum_{i=1}^{t} (r_i \cdot a_i) \odot_d s \\ &= \lfloor q/p \rfloor \cdot m + \sum_{i=1}^{t} r_i \odot_d e_i \in \mathbb{Z}_q^d[x] \approx \lfloor q/p \rfloor \cdot m, \end{aligned}$$

using **'associative' property** of middle-product:

$$r_i \odot_d (a_i \odot_{d+k} s) = (r_i \cdot a_i) \odot_d s$$

Compute tight upper bound on decryption error prob. p_e (Hoeffding)

Specified Parameters

- We specify total of 6 different parameters sets Toy64, Lite96, Std128, Med160, Hi192, Super256
  - last digits = equivalent symmetric key search security level
- Achieve NIST security goals at high quantum key search security goals (MAXDEPTHMD $= 2^{40}$)
  - Std128 - NIST level 1 (AES-128): goal $\lambda_C = 143$, $\lambda_Q = 130$
  - Hi192 - NIST level 3 (AES-192): goal $\lambda_C = 207$, $\lambda_Q = 193$
  - Super256 - NIST level 5 (AES-256): $\lambda_C = 272$, $\lambda_Q = 258$

Titanium-CPA Parameters

Table: Determined Titanium-CPA core parameters.

Parameter	Toy64	Lite96	Std128	Med160	Hi192	Super256
n	684	800	1024	1280	1536	2048
k	255	479	511	511	767	1023
d	256	256	256	256	256	256
t	10	8	9	9	7	7
q	240641	84481	86017	301057	737281	1198081
p	2	2	2	2	2	2
cmp	10	9	9	11	12	13

Titanium-CCA Parameters

Table: Determined Titanium-CCA core parameters.

Parameter	Toy64	Lite96	Std128	Med160	Hi192	Super256
n	684	800	1024	1280	1536	2048
k	255	479	511	511	767	1023
d	256	256	256	256	256	256
t	10	9	10	10	8	8
q	471041	115201	118273	430081	783361	1198081
p	2	2	2	2	2	2
cmp	11	9	9	11	12	13

Design Rationale

- **Choice of Error distributions:**
  1. Secret key: Uniform distrib. coeffs over $\mathbb{Z}_q$
     - sample directly in the NTT domain from seedsk (save NTT)
  2. Uniform distrib. over $[-2^b, 2^b]$ for encryption randomness coeff.
     - Uniform shape: max. min-entropy (LHL) for given variance (dec. error probability)
     - Size of b: optimize to reduce pk+ciph size
     - Power of 2: efficient sampling
     - Fine tweak: two int. values of b for two subsets of r_i coeffs.
  3. 'Binomial Difference' distribution for errors = Bin$(4, 1/2)$ - Bin$(4, 1/2)$
     - std. dev. of error coeff. $= \sqrt{2}$, fast constant-time sampling
     - $\approx$ Gaussian shape as in worst-case hardness proofs

Design Rationale

- **Choice of Error distributions:**
  1. Secret key: Uniform distrib. coeffs over $\mathbb{Z}_q$
     - sample directly in the NTT domain from seedsk (save NTT)
  2. Uniform distrib. over $[-2^b, 2^b]$ for encryption randomness coeff.
     - Uniform shape: max. min-entropy (LHL) for given variance (dec. error probability)
     - Size of b: optimize to reduce pk+ciph size
     - Power of 2: efficient sampling
     - Fine tweak: two int. values of b for two subsets of r_i coeffs.
  3. 'Binomial Difference' distribution for errors = Bin(4, 1/2) - Bin(4, 1/2)
     - std. dev. of error coeff. $= \sqrt{2}$, fast constant-time sampling
     - $\approx$ Gaussian shape as in worst-case hardness proofs

- **Decryption error probability p_e:**
  1. A moderate goal $p_e = 2^{-30}$ for Titanium-CPA, and
  2. Set to a cryptographically negligible value for Titanium-CCA (provably avoid decryption failure attacks)

Optimized Implementation

- **Fast Middle Product Algorithm and Optimisations:**
  1. Middle product NTT-based algorithm,
     - 3 NTT dims needed: $\mathrm{d}_1 \geq d + k, \mathrm{d}_2 \geq n + k, \mathrm{d}_3 \geq n + d + k - 1$
     - **Choice of NTT dims:** small multiples of 256
     - Core NTT = radix 2 algorithm in dim. 256
     - **Choice of parameters** k, n**:** close to multiples of 256 (min. pad)
     - **Choice of** q**:** 'NTT-friendly' prime wrt $\mathrm{d}_1, \mathrm{d}_2, \mathrm{d}_3$.
     - Partial MP-NTT: exploit MP truncation, input padding
     - Fast mod q reduction (Barret and Mongomery),
  2. Optimised Titanium-CPA and Titanium-CCA Algorithms:
     - Precompute pub-key NTT in keygen. (save NTT from enc and CCA dec)
     - Sample secret key directly in NTT domain (save NTT)
  3. Constant-time implementation:

Optimized Implementation

- **Fast Middle Product Algorithm and Optimisations:**
  1. Middle product NTT-based algorithm,
     - 3 NTT dims needed: $\mathrm{d}_1 \geq d + k, \mathrm{d}_2 \geq n + k, \mathrm{d}_3 \geq n + d + k - 1$
     - **Choice of NTT dims:** small multiples of 256
     - Core NTT = radix 2 algorithm in dim. 256
     - **Choice of parameters** k, n**:** close to multiples of 256 (min. pad)
     - **Choice of** q**:** 'NTT-friendly' prime wrt $\mathrm{d}_1, \mathrm{d}_2, \mathrm{d}_3$.
     - Partial MP-NTT: exploit MP truncation, input padding
     - Fast mod q reduction (Barret and Mongomery),
  2. Optimised Titanium-CPA and Titanium-CCA Algorithms:
     - Precompute pub-key NTT in keygen. (save NTT from enc and CCA dec)
     - Sample secret key directly in NTT domain (save NTT)
  3. Constant-time implementation:

- **Additional optimized implementations:**
  1. Intel AVX2 instruction set

Titanium Security Analysis

Security of Titanium: Approach

Main security analysis approach:

- **Concrete security proof** from hardest PLWE^f over f in family $\mathcal{F}$:
  - Part 1: Security of Titanium-CPA/Titanium-CCA from hardness of MP-LWE
  - Part 2: MP-LWE hardness from PLWE^f hardness over many f's
    - Already discussed in 'security foundations'
- **Use proof bounds to select parameters:** low bound for security of Titanium-CPA/Titanium-CCA, assuming
  - **best known dual BKZ attack on** PLWE^f (any f in family $\mathcal{F}$)
  - **conservative 'Core SVP' security estimate** for dual BKZ attack [ADPS16]
  - We followed **more conservative/higher safety margins** (bigger than some other lattice-based proposals)

Part 1: Titanium-CPA security from hardness of MP-LWE

Theorem (IND-CPA of Titanium-CPA from MP-LWE)

Assume q is prime and Leftover Hash Lemma (LHL) condition holds:

$$t \cdot (k+1) \cdot b_{\mathsf{LHL}} \geq 2 \cdot (\log(\Delta_{\mathsf{LHL}}^{-1}) - 1) + (n + d + k) \cdot \log q. \quad (1)$$

Then, any IND-CPA *attack against* Titanium-CPA *with run-time T and advantage ε in the (classical) Random Oracle Model for XOF (Q queries), implies an* IND-CPA *attack against* MP-LWE$^{n}_{q,\alpha,\chi,d'=d+k}$ *with run-time $T' \approx T$, and distinguishing advantage*

$$\varepsilon' \geq \varepsilon/2 - 3 \cdot Q/2^{256} - \Delta_{\mathsf{LHL}}. \quad (2)$$

- b_{LHL} = bit length of encryption random polynomials r_i
- XOF = hash function used to derive randomness from short seeds
- Δ_{LHL} chosen as $O(2^{-\lambda})$ for security parameter λ

Proof idea

Recall Titanium-CPA ciphertext form

$$c_1 = \sum_{1 \le i \le t} r_i \cdot a_i \text{ and } c_2 = \sum_{1 \le i \le t} r_i \odot_d b_i + m \cdot \lfloor q/2 \rfloor,$$

using random r_i's with 'small' coefficients each with entropy b_{LHL}.

Security argument ([RSSS17], variant of 'primal Regev'):

- Replace in key gen. $(a_i \hookleftarrow U(\mathbb{Z}_q^{<n}[x]), b_i = a_i \odot_{d+k} s + e_i)$ with uniformly pairs $(a_i \hookleftarrow U(\mathbb{Z}_q^{<n}[x]), b_i \hookleftarrow U(\mathbb{Z}_q^{<d+k}[x]))$.
  - MP-LWE$^n_{q,\alpha,\chi,d'=d+k}$ hardness implies attacker's view stays comp. indistinguishable
- Now, in challenge c_2, $\sum_{1 \le i \le t} r_i \odot_d b_i$ is stat. indistinguishable from uniform on $\mathbb{Z}_q^{<d}[x]$ (given pub key and c_1), stat. masks message m
  - Implied by 'generalized' Leftover Hash Lemma (LHL) if q prime and min-entropy of r_i's sufficiently exceeds max-entropy of ciphertext space
  - c_1 = 'auxilliary information' on r_i's, not uniform (no security impact)

Part 1: Titanium-CCA security from hardness of MP-LWE

Theorem (IND-CCA of Titanium-CCA from MP-LWE)

Assume q is prime, LHL condition holds and Titanium-CPA *is p_e-correct. Then, any IND-CCA attack against* Titanium-CCA *with run-time T and advantage ε with at most $Q_{\mathsf{XOF}}, Q_{\mathsf{G}}, Q_{\mathsf{H}}$ queries in the (classical) Random Oracle Model for* XOF, G *and* H *respectively, implies an attack against* $\text{MP-LWE}^{n}_{q,\alpha,\chi,d'=d+k}$ *with run-time $T' \approx T$, and distinguishing advantage*

$$\varepsilon' \geq \frac{1}{6} \cdot \left(\varepsilon - Q_{\mathsf{G}} \cdot p_e - \Delta_{\mathsf{LHL}} - \frac{10 \cdot Q_{\mathsf{XOF}} + 2 \cdot Q_{\mathsf{G}} + Q_{\mathsf{H}} + 1}{2^{256}} \right). \quad (3)$$

- Tight proof by combining Titanium-CPA proof with tight Fujisako-Okamoto transform proof [HHK17]
- p_e set to $O(2^{-\lambda})$ - provably avoid decryption fail attacks
- Classical ROM could be replaced by quantum ROM [HHK17] (but with non-tight security reduction)

How we set the Titanium-CPA/Titanium-CCA parameters

Def. of quantum (classical) security levels λ_Q (λ_C): For any attack with time $T_Q \leq 2^{\lambda_Q}$ ($T_C \leq 2^{\lambda_C}$) and advantage ε_Q (ε_C), we have

$$T_Q/\varepsilon_Q \geq 2^{\lambda_Q} \qquad (T_C/\varepsilon_C \geq 2^{\lambda_C}).$$

- Time = no. of elem. quantum (classical) gates
- RO query cost = $T_{QRO} \approx 2^{19}$ qu. ($T_{RO} \approx 2^{19}$ cl.) gates
- RO query quantum depth = $D_{QRO} = 2^{13}$ qu. gates

Main parameter selection goal: Set parameters (using security proof) to get **proven** 2^{λ_C} classical security level for Titanium-CPA/Titanium-CCA, assuming:

- Hardest PLWE^f security level ($f \in \mathcal{F}$) = sec. level of best known PLWE^f attack (dual lattice attack)
- Classical random oracle model for symmetric key-based functions H,G,XOF
- Conservative 'core SVP' methodology [ADPS16] to estimate dual lattice attack complexity level

How we set the Titanium-CPA/Titanium-CCA parameters

Quantum security estimate approach:

- **Problem:** Existing **quantum** random oracle model security proofs for Titanium-CPA/Titanium-CCA are not tight
- **Approach:** Modify classical bounds to account for Grover search bounds:
  - Replace classical 'bad event prob.' terms $Q \cdot \delta$, in classical proof:

$$p_C \leq Q \cdot \delta$$

  - by quantum 'Grover-search' bounds with Q/Q_D parallel Grover search circuits of depth Q_D queries :

$$p_Q \leq 8 \cdot (Q/Q_D) \cdot Q_D^2 \cdot \delta$$

  - **Max Quantum Depth (MD) constraint:** $\mathrm{MD} = 2^{40}, 2^{64}, 2^{96}$ gates: we satisfy security goals at all these MD values.
  - Maximum number of queries in quantum Grover search $Q_D = \min(\mathrm{MD}, 2^{\lambda_Q})/\mathrm{D}_{QRO}$

Parameter Setting: Conservative assumptions

Our conservative parameter setting assumptions:

- Include security proof reduction costs in parameter selection
- Extra safety margins for future cryptanalytic progress:
  - 10% safety margin on quantum bit security level
- **Use** minimum f degree m_{min} in $\mathcal{F}$ for PLWE^f hardness estimates
- 'Core SVP' approach for PLWE^f dual attack BKZ cost - leave room for future cryptanalytic progress:
  - lower bound gate complexity of BKZ-b by $T = 2^{0.292 \cdot b}$ (resp. $2^{0.265 \cdot b}$)
  - **don't rely on additional costs** related to
    - no. of SVP calls of BKZ, memory access costs, Grover iteration costs
    - **remark:** some proposals assume costs: harder to compare proposals.
    - **Q:** Could NIST recommend a standard cost measure for BKZ?
  - assume each sieve SVP call provides up T (not only M) short vectors
  - unlimited quantum circuit depth for SVP sieve
- **Don't rely on** MP-LWE **being harder than** PLWE^f
  - although best known attack on MP-LWE is significantly harder

Claimed Security Levels: Titanium-CPA/Titanium-CCA

Scheme	Param.	MD = 40		MD = 64		MD = 96	
		Goal	Min. Cl.	Goal	Min. Cl.	Goal	Min. Cl.
Titanium-CCA-Toy64	λ_Q	66	73	53	61	53	61
Titanium-CCA-Toy64	λ_C	79	82	79	82	79	82
Titanium-CPA-Toy64	λ_Q	66	83	53	83	53	83
Titanium-CPA-Toy64	λ_C	79	90	79	90	79	90
Titanium-CCA-Lite96	λ_Q	98	110	74	110	69	110
Titanium-CCA-Lite96	λ_C	111	126	111	120	111	120
Titanium-CPA-Lite96	λ_Q	98	115	74	115	69	115
Titanium-CPA-Lite96	λ_C	111	126	111	126	111	126
Titanium-CCA-Std128	λ_Q	130	134	106	126	85	105
Titanium-CCA-Std128	λ_C	143	146	143	146	143	146
Titanium-CPA-Std128	λ_Q	130	155	106	159	85	159
Titanium-CPA-Std128	λ_C	143	164	143	164	143	164
Titanium-CCA-Med160	λ_Q	162	176	138	164	106	132
Titanium-CCA-Med160	λ_C	175	192	175	192	175	192
Titanium-CPA-Med160	λ_Q	162	183	138	187	106	186
Titanium-CPA-Med160	λ_C	175	199	175	200	175	200
Titanium-CCA-Hi192	λ_Q	193	207	169	183	137	151
Titanium-CCA-Hi192	λ_C	207	230	207	230	207	230
Titanium-CPA-Hi192	λ_Q	193	214	169	217	137	187
Titanium-CPA-Hi192	λ_C	207	234	207	237	207	237
Titanium-CCA-Super256	λ_Q	258	240	234	216	202	184
Titanium-CCA-Super256	λ_C	272	266	272	266	272	266
Titanium-CPA-Super256	λ_Q	258	243	234	219	202	187
Titanium-CPA-Super256	λ_C	272	269	272	269	272	269

Best known attacks on Titanium-CPA/Titanium-CCA: is MP-LWE harder than PLWE^f?

Our proof shows MP-LWE$^n_{q,\alpha,\chi,d'}$ is **at least** as hard as PLWE$^f_{q,\alpha,\chi}$.
Is it actually **harder** for small no. of MP-LWE samples t?

Best known attack on MP-LWE$^n_{q,\alpha,\chi,d'}$ has higher complexity than PLWE$^f_{q,\alpha,\chi}$:

- Generic LWE attack on MP-LWE$^n_{q,\alpha,\chi,d'}$ uses secret in dimension $n + d'$, versus $\leq n$ for PLWE$^f_{q,\alpha,\chi}$
- We give an MPLWE-optimised LWE attack to reduce secret dimension down to $n + m/t \approx n$ for m LWE samples:
  - **Idea:** Exploit the zeros in the Toeplitz matrix for a_i's: Keep only m/t top rows of each Toeplitz matrix
  - Still leaves a hardness gap of $q^{1/t}$ in approx.-SVP factor to best known attack on PLWE$^f_{q,\alpha,\chi}$ with f in dimension n

Best known attacks on Titanium-CPA/Titanium-CCA: MP-LWE Complexity estimates

'Core-SVP' Complexity of MPLWE-optimised primal embedding LWE attack on MP-LWE$^{n}_{q,\alpha,\chi,d'}$ $(\lambda_{C,\text{emb},2}/\lambda_{C,\text{emb},2})$, compared to PLWE$^{f}_{q,\alpha,\chi}$ with f in dimension n $(\lambda_{C,\text{PLWE},m_{\max}}/\lambda_{C,\text{PLWE},m_{\max}})$ and scheme goals (λ_C/λ_Q)

Par. Set	**Classical**			**Quantum**		
	$\lambda_{C,\text{emb},2}$	$\lambda_{C,\text{PLWE},m_{\max}}$	λ_C	$\lambda_{Q,\text{emb},2}$	$\lambda_{Q,\text{PLWE},m_{\max}}$	λ_Q
CCA, Toy64	125	90	79	113	83	66
CPA, Toy64	134	97	79	121	89	66
CCA, Lite96	181	129	111	164	118	98
CPA, Lite96	194	133	111	176	122	98
CCA, Std128	236	176	143	214	161	130
CPA, Std128	251	182	143	228	166	130
CCA, Med160	274	205	175	248	187	162
CPA, Med160	291	211	175	264	194	162
CCA, Hi192	345	243	207	313	222	193
CPA, Hi192	363	244	207	330	224	193
CCA, Super256	467	333	272	424	305	258
CPA, Super256	489	333	272	444	305	258

Best known attacks on Titanium-CPA/Titanium-CCA: Complexity estimates

Best known attack complexity (MP-LWE Core-SVP or Brute force/Grover) on Titanium-CPA/Titanium-CCA

Par. Set	Classical			Quantum		
	$\lambda_{C,\text{bstatk}}$	$\lambda_{C,\text{PLWE},m_{\max}}$	λ_C	$\lambda_{Q,\text{bstatk}}$	$\lambda_{Q,\text{PLWE},m_{\max}}$	λ_Q
CCA, Toy64	125	90	79	113	83	66
CPA, Toy64	134	97	79	121	89	66
CCA, Lite96	181	129	111	164	118	98
CPA, Lite96	194	133	111	176	122	98
CCA, Std128	236	176	143	214	161	130
CPA, Std128	251	182	143	228	166	130
CCA, Med160	272	205	175	245	187	162
CPA, Med160	272	211	175	245	194	162
CCA, Hi192	272	243	207	245	222	193
CPA, Hi192	272	244	207	245	224	193
CCA, Super256	272	333	272	245	305	258
CPA, Super256	272	333	272	245	305	258

Conservative Parameter Comparison

Comparison of best known LWE/PLWE^f attacks complexity:

$\lambda_{Q,\text{LWE},Al,du}/\lambda_{Q,\text{LWE},Al,pr}$ = LWE/PLWE^f dual/primal attack complexity via [Albrecht et al.] LWE Estimator (Q-core-Sieve model)

$m = n$ for Titanium (**highest** degree polynomials in $\mathcal{F}$)

LWE security for NIST Level 1 schemes (AES128 eq. security)

Par. Set	$\lambda_{Q,\text{LWE},Al,du}$	$\lambda_{Q,\text{LWE},Al,pr}$
Titanium – Std128	195	168
FrodoKEM – 640	159	129
NewHope – 1024	137	103
Kyber – 512	137	103

Performance Summary

Benchmark results for Titanium-CPA.

Benchmarks for our updated Titanium-CPA Imp. (see 'Imp. Updates')

Par. Set	Number of cycles (no AVX2)	Number of cycles (AVX2)
Toy64	KeyGen: 1264647 Encrypt: 900120 Decrypt: 152705	KeyGen: 692914 Encrypt: 525372 Decrypt: 87809
Lite96	KeyGen: 1269479 Encrypt: 1073167 Decrypt: 183832	KeyGen: 710227 Encrypt: 606443 Decrypt: 102122
Std128	KeyGen: 1619550 Encrypt: 1262047 Decrypt: 217612	KeyGen: 828566 Encrypt: 742669 Decrypt: 119554
Med160	KeyGen: 1877257 Encrypt: 1646486 Decrypt: 253458	KeyGen: 1069528 Encrypt: 957079 Decrypt: 140183
Hi192	KeyGen: 1894719 Encrypt: 1763250 Decrypt: 323977	KeyGen: 1060605 Encrypt: 992393 Decrypt: 188572
Super256	KeyGen: 2486436 Encrypt: 2450834 Decrypt: 439522	KeyGen: 1319663 Encrypt: 1437211 Decrypt: 230179

Benchmark results for Titanium-CCA.

Benchmarks for our updated Titanium-CCA Imp. (see 'Imp. Updates')

Par. Set	Number of cycles (no AVX2)	Number of cycles (AVX2)
Toy64	KeyGen: 1269090	KeyGen: 702889
	Encrypt: 947906	Encrypt: 561925
	Decrypt: 1107424	Decrypt: 678669
Lite96	KeyGen: 1426439	KeyGen: 777025
	Encrypt: 1234901	Encrypt: 682349
	Decrypt: 1425403	Decrypt: 817124
Std128	KeyGen: 1806119	KeyGen: 931563
	Encrypt: 1446751	Encrypt: 867475
	Decrypt: 1671578	Decrypt: 1029815
Med160	KeyGen: 2035675	KeyGen: 1155742
	Encrypt: 1855415	Encrypt: 1079706
	Decrypt: 2109199	Decrypt: 1186795
Hi192	KeyGen: 2122547	KeyGen: 1224984
	Encrypt: 1986198	Encrypt: 1154631
	Decrypt: 2310815	Decrypt: 1309829
Super256	KeyGen: 2829289	KeyGen: 1432721
	Encrypt: 2799390	Encrypt: 1574260
	Decrypt: 3247542	Decrypt: 1879128

Key/Ciphertext Lengths

Par. Set	Titanium-CPA Length (byte)	Titanium-CCA Length (byte)
Toy64	pk: 11552 sk: 32 ct: 2560	pk: 12192 sk: 12224 / sk*: 32 ct: 2720
Lite96	pk: 13088 sk: 32 ct: 2976	pk: 14720 sk: 14752 / / sk*: 32 ct: 3008
Std128	pk: 14720 sk: 32 ct: 3520	pk: 16352 sk: 16384 / sk*: 32 ct: 3552
Med160	pk: 16448 sk: 32 ct: 4512	pk: 18272 sk: 18304 / / sk*: 32 ct: 4544
Hi192	pk: 17952 sk: 32 ct: 6016	pk: 20512 sk: 20544 / / sk*: 32 ct: 6048
Super256	pk: 23552 sk: 32 ct: 8320	pk: 26912 sk: 26944 / / sk*: 32 ct: 8352

Performance comparison with some other proposals

Prelim. Comparison with some NIST Level 1 proposals:

- Large time gain over FrodoKEM without AES-NI instructions
- Does **not** take into account larger PLWE^f security margin for Titanium

Scheme	Hard Problem	$\mathcal{F}$ Size	Eff. Aspects Size (Bytes)	Time (Cycles)
Kyber-512	Module PLWE^f	1	$\|pk\| = 736$ $\|sk\| = 1632$ $\|ct\| = 800$	K : 141872 E : 205468 D : 246040
Titanium-CCA-Std128	MP-LWE	$\geq 3^{256}$	$\|pk\| = 16352$ $\|sk\| = 16384$ $\|ct\| = 3552$	K : 1806119 E : 1446751 D : 1671578
FrodoKEM-640 – *cSHAKE*	LWE	n/a	$\|pk\| = 9616$ $\|sk\| = 19872$ $\|ct\| = 9736$	K : 8297000 E : 9082000 D : 9077000

Performance comparison with some other proposals

- Noticeable time gain over FrodoKEM **with AES-NI instructions**
- Does **not** take into account larger PLWE^f security margin for Titanium

Scheme	Hard Problem	$\mathcal{F}$ Size	Eff. Aspects Size (Bytes)	Time (Cycles)
Titanium-CCA-Std128-AES	MP-LWE	$\geq 3^{256}$	$\|pk\| = 16352$ $\|sk\| = 16384$ $\|ct\| = 3552$	K : 1553925 E : 1248256 D : 1439221
FrodoKEM-640 – *AES*	LWE	n/a	$\|pk\| = 9616$ $\|sk\| = 19872$ $\|ct\| = 9736$	K : 1287000 E : 1810000 D : 1811000

Titanium-CCA-Std128-AES: Titanium-CCA with AES PRG (new implementation, not in NIST sub.)

Implementation Updates after NIST submission

Implementation updates

- NIST implementation may not be constant time depending on C compiler imp of % mod reduction
  - Rewrote mod reduction to avoid % compiler independent constant-time
- Improved efficiency of NTT implementation
  - by merging intermediate levels of radix-2 NTT (mod red at end only)
- Added OpenQuantum integration for Titanium
- New AES-based PRG Titanium variant Titanium – AES (not in NIST submission)
  - Faster symmetric-key XOF for a_i, sk gen. using Intel AES-NI instructions
- Updated Implementation benchmarks

Benchmark results of the Titanium-CPA with AES-NI.

Par. Set	Number of cycles (no AVX2)	Number of cycles (AVX2)
Toy64	KeyGen: 1080756 Encrypt: 751720 Decrypt: 132875	KeyGen: 426869 Encrypt: 315424 Decrypt: 54228
Lite96	KeyGen: 1099505 Encrypt: 935097 Decrypt: 159962	KeyGen: 436896 Encrypt: 383282 Decrypt: 65568
Std128	KeyGen: 1396315 Encrypt: 1079362 Decrypt: 193412	KeyGen: 501206 Encrypt: 473929 Decrypt: 73888
Med160	KeyGen: 1612734 Encrypt: 1436597 Decrypt: 221925	KeyGen: 639902 Encrypt: 590207 Decrypt: 91578
Hi192	KeyGen: 1631230 Encrypt: 1530741 Decrypt: 286266	KeyGen: 655928 Encrypt: 632895 Decrypt: 119864
Super256	KeyGen: 2185642 Encrypt: 2182793 Decrypt: 395959	KeyGen: 788307 Encrypt: 934890 Decrypt: 155303

Benchmark results of the Titanium-CCA with AES-NI.

Par. Set	Number of cycles (no AVX2)	Number of cycles (AVX2)
Toy64	KeyGen: 1085654	KeyGen: 430751
	Encrypt: 801691	Encrypt: 347802
	Decrypt: 937381	Decrypt: 400762
Lite96	KeyGen: 1233880	KeyGen: 488751
	Encrypt: 1085114	Encrypt: 458299
	Decrypt: 1248694	Decrypt: 522409
Std128	KeyGen: 1553925	KeyGen: 556385
	Encrypt: 1248256	Encrypt: 555743
	Decrypt: 1439221	Decrypt: 627455
Med160	KeyGen: 1775307	KeyGen: 704882
	Encrypt: 1638219	Encrypt: 695653
	Decrypt: 1863957	Decrypt: 785317
Hi192	KeyGen: 1843318	KeyGen: 740083
	Encrypt: 1760607	Encrypt: 755662
	Decrypt: 2041340	Decrypt: 876249
Super256	KeyGen: 2493796	KeyGen: 895571
	Encrypt: 2521849	Encrypt: 1123224
	Decrypt: 2919891	Decrypt: 1276374

Thank you.