Toward New Practical Digital Signature Scheme based on Lattices

Duhyeong Kim
Joint work with
Jung Hee Cheon, Jaeyoon Kim, Yongha Son

Crypto Winter Workshop
Jan 03, 2019

Hash-and-Sign

Hash-and-Sign Signature

Basic Framework

Hash-and-Sign Signature

Basic Framework

- Given a trapdoor lattice A (VK) with a trapdoor T (SK)

Hash-and-Sign Signature

Basic Framework

- Given a trapdoor lattice A (VK) with a trapdoor T (SK)
- For a message m, generate s (Sign) satisfying

$$As = H(m) \ \& \ \|s\| < \beta$$

Hash-and-Sign Signature

Basic Framework

- Given a trapdoor lattice A (VK) with a trapdoor T (SK)
- For a message m, generate s (Sign) satisfying

$$As = H(m) \ \& \ \boxed{\|s\| < \beta} \ (\Leftarrow \text{via the trapdoor } T)$$

Hash-and-Sign Signature

Basic Framework

- Given a trapdoor lattice A (VK) with a trapdoor T (SK)
- For a message m, generate s (Sign) satisfying

$$As = H(m) \ \& \ \boxed{\|s\| < \beta} \ (\Leftarrow \text{via the trapdoor } T)$$

- Security
  - Hard to find SK from VK and pairs of (m, s)

Hash-and-Sign Signature

Basic Framework

- Given a trapdoor lattice A (VK) with a trapdoor T (SK)
- For a message m, generate s (Sign) satisfying

$$As = H(m) \ \& \ \boxed{\|s\| < \beta} \ (\Leftarrow \text{via the trapdoor } T)$$

- Security
  - Hard to find SK from VK and pairs of (m, s)
  - Hard to find "small" s satisfying $As = H(m)$

Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

- Gentry-Peikert-Vaikuntanathan [GPV08]

Hash-and-Sign Signature

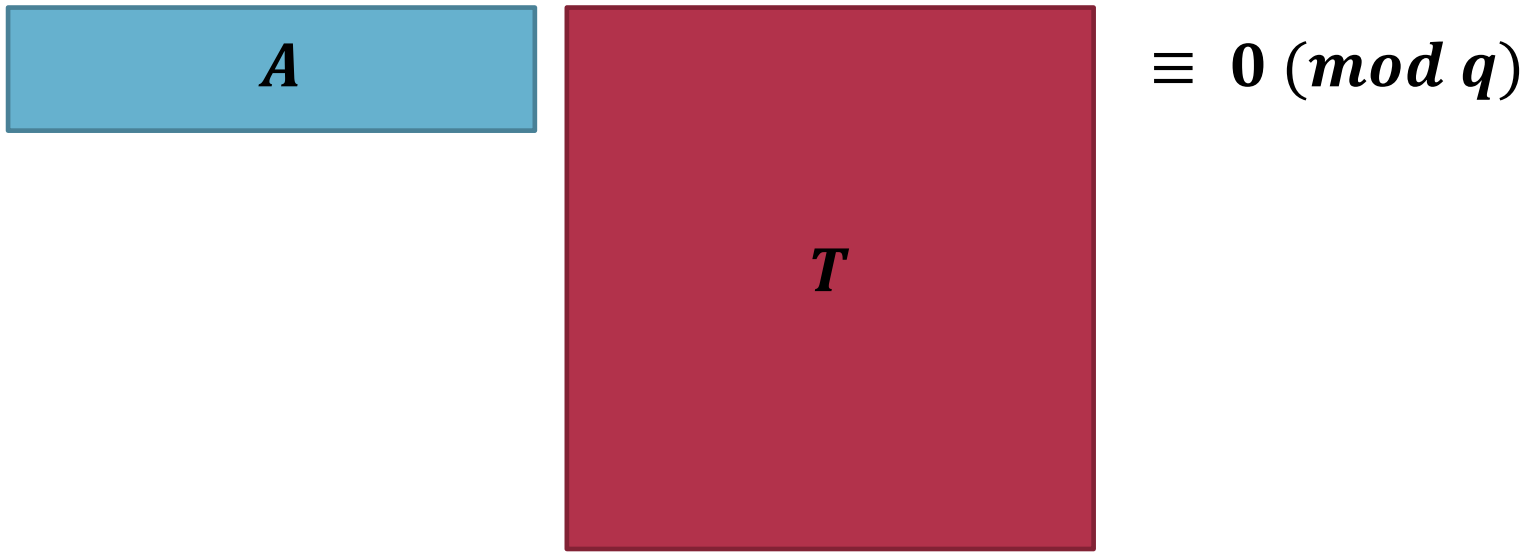
How to generate a Trapdoor Lattice?

- Gentry-Peikert-Vaikuntanathan [GPV08]
  - For $q = poly(n)$ and $m = \omega(n \log q)$, there exists an algorithm outputs $(A, T) \in Z_q^{n \times m} \times Z_q^{m \times m}$ s.t.

Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

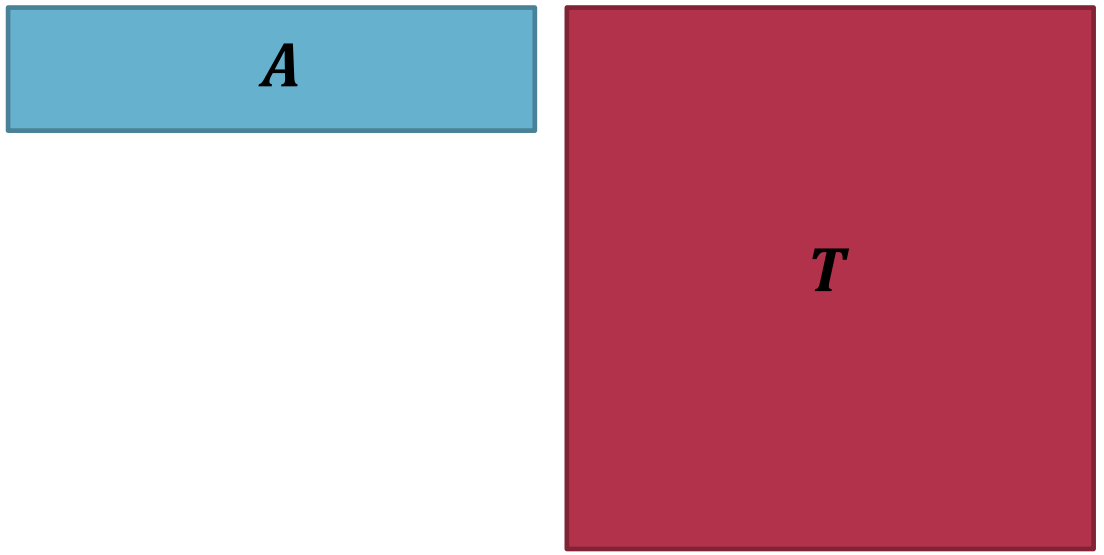
- Gentry-Peikert-Vaikuntanathan [GPV08]
  - For $q = poly(n)$ and $m = \omega(n \log q)$, there exists an algorithm outputs $(A, T) \in Z_q^{n \times m} \times Z_q^{m \times m}$ s.t.



Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

- Gentry-Peikert-Vaikuntanathan [GPV08]
  - For $q = poly(n)$ and $m = \omega(n \log q)$, there exists an algorithm outputs $(A, T) \in Z_q^{n \times m} \times Z_q^{m \times m}$ s.t.



$\equiv \mathbf{0}\ (\boldsymbol{mod\ q})$

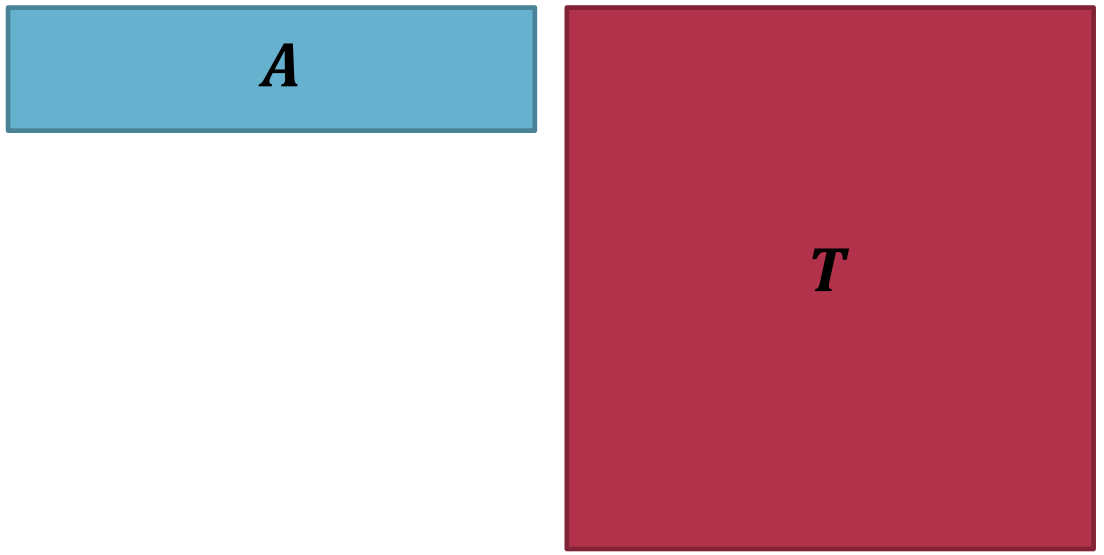
satisfying

1. $\|T\| \leq m^{1+\epsilon}$ (a basis of $\Lambda_q^{\perp}(A)$)

Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

- Gentry-Peikert-Vaikuntanathan [GPV08]
  - For $q = poly(n)$ and $m = \omega(n \log q)$, there exists an algorithm outputs $(A, T) \in Z_q^{n \times m} \times Z_q^{m \times m}$ s.t.



$\equiv \mathbf{0}\ (\boldsymbol{mod\ q})$

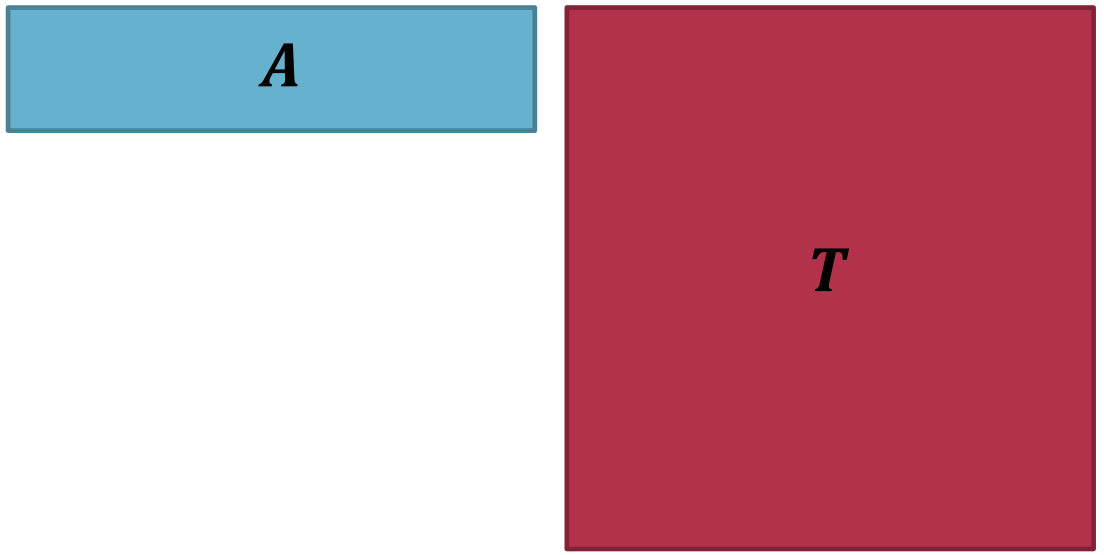
satisfying

1. $\|T\| \leq m^{1+\epsilon}$ (a basis of $\Lambda_q^{\perp}(A)$)
2. $D(A) \approx U(Z_q^{n \times m})$

Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

- Gentry-Peikert-Vaikuntanathan [GPV08]
  - For $q = poly(n)$ and $m = \omega(n \log q)$, there exists an algorithm outputs $(A, T) \in Z_q^{n \times m} \times Z_q^{m \times m}$ s.t.



$$\equiv \mathbf{0} \ (\boldsymbol{mod\ q})$$

satisfying

1. $\|T\| \leq m^{1+\epsilon}$ (a basis of $\Lambda_q^{\perp}(A)$)
2. $D(A) \approx U(Z_q^{n \times m})$

  - Drawback in Practicality: too large parameters (due to statistical property and w/o ring structure)

Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

- Ducas-Lyubashevsky-Prest [DLP14]

Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

- Ducas-Lyubashevsky-Prest [DLP14]
  - Substitute a plain trapdoor lattice w/ statistical property by a ring structure and computational property

Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

- Ducas-Lyubashevsky-Prest [DLP14]
  - Substitute a plain trapdoor lattice w/ statistical property by a ring structure and computational property

"NTRU Trapdoor Lattice"

Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

- Ducas-Lyubashevsky-Prest [DLP14]
  - Substitute a plain trapdoor lattice w/ statistical property by a ring structure and computational property

"NTRU Trapdoor Lattice"

  - For "small" $f, g \in R_q = Z_q[X]/(X^n + 1)$ and $h = \frac{g}{f}$, one can generate "small" $F, G \in R_q$ s.t.

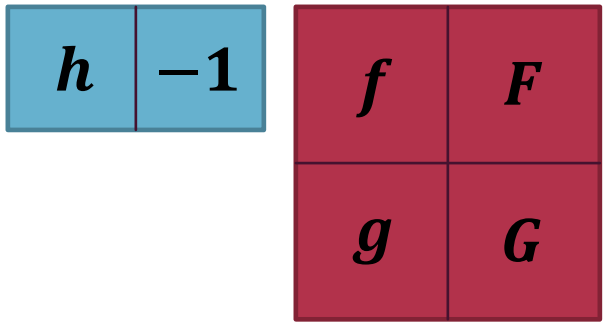
Hash-and-Sign Signature

How to generate a Trapdoor Lattice?

- Ducas-Lyubashevsky-Prest [DLP14]
  - Substitute a plain trapdoor lattice w/ statistical property by a ring structure and computational property

"NTRU Trapdoor Lattice"

  - For "small" $f, g \in R_q = Z_q[X]/(X^n + 1)$ and $h = \frac{g}{f}$, one can generate "small" $F, G \in R_q$ s.t.



$$\equiv \mathbf{0}\ (\boldsymbol{mod}\ \boldsymbol{q})$$

satisfying

$$fG - gF = q$$

High-level structure of Trapdoor Lattice

High-level structure of Trapdoor Lattice

Our Goal:

To publish a (ring) lattice $A \in R_q^{k \times d}$ w/ short basis T of $\Lambda_q^{\perp}(A)$, which does NOT give any information of T computationally

High-level structure of Trapdoor Lattice

Our Goal:

To publish a (ring) lattice $A \in R_q^{k \times d}$ w/ short basis T of $\Lambda_q^{\perp}(A)$, which does NOT give any information of T computationally

"How can we interpret the NTRU trapdoor lattice in high-level?"

High-level structure of Trapdoor Lattice

Observations

High-level structure of Trapdoor Lattice

Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!

High-level structure of Trapdoor Lattice

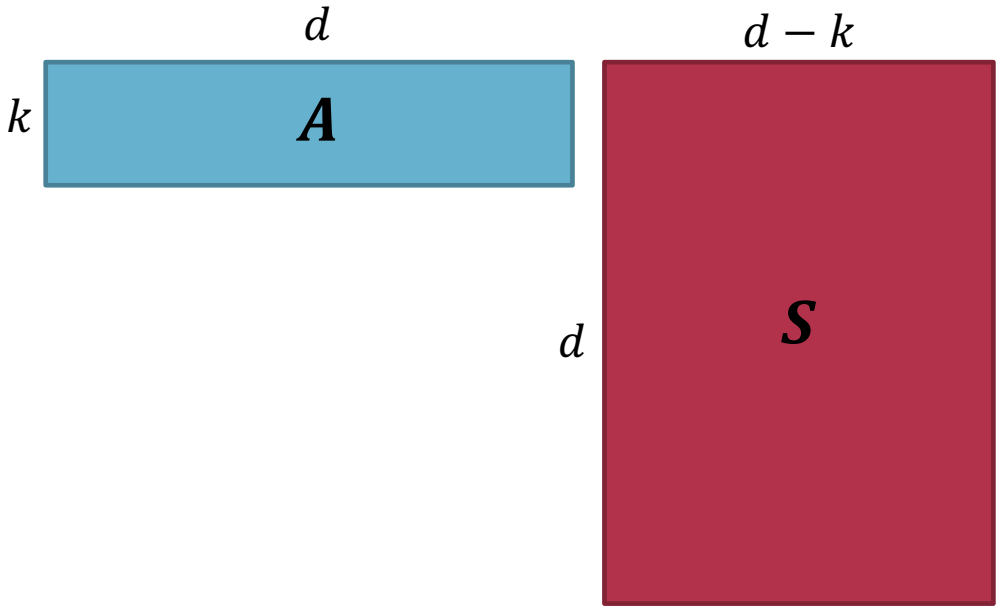
Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!
- Rank(T) over R_q < Rank(T) over R $\Longrightarrow$ we can NOT arbitrarily choose all elements of T

High-level structure of Trapdoor Lattice

Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!
- Rank(T) over R_q < Rank(T) over R $\Longrightarrow$ we can NOT arbitrarily choose all elements of T



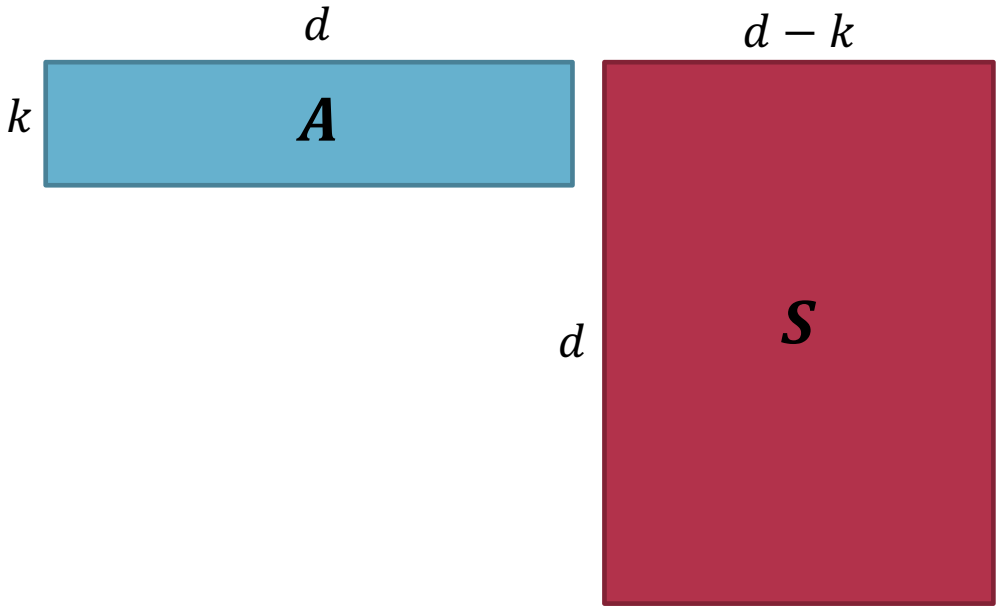
$$\equiv \mathbf{0}\ (\boldsymbol{mod}\ \boldsymbol{q})$$

- $d - k$ = Rank(T) over R_q < d

High-level structure of Trapdoor Lattice

Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!
- Rank(T) over R_q < Rank(T) over R $\Longrightarrow$ we can NOT arbitrarily choose all elements of T



$$\equiv \mathbf{0}\ (\boldsymbol{mod\ q})$$

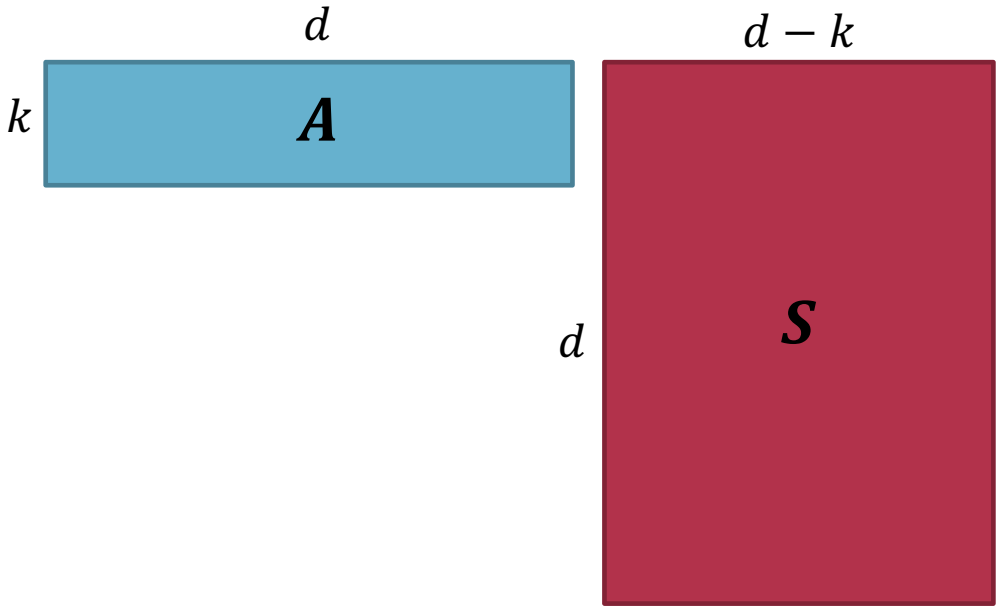
❖ $d - k$ = Rank(T) over $R_q < d$

1. Choose short $S \in R_q^{d \times (d-k)}$

High-level structure of Trapdoor Lattice

Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!
- Rank(T) over R_q < Rank(T) over R $\Longrightarrow$ we can NOT arbitrarily choose all elements of T



$$\equiv \mathbf{0}\ (\boldsymbol{mod\ q})$$

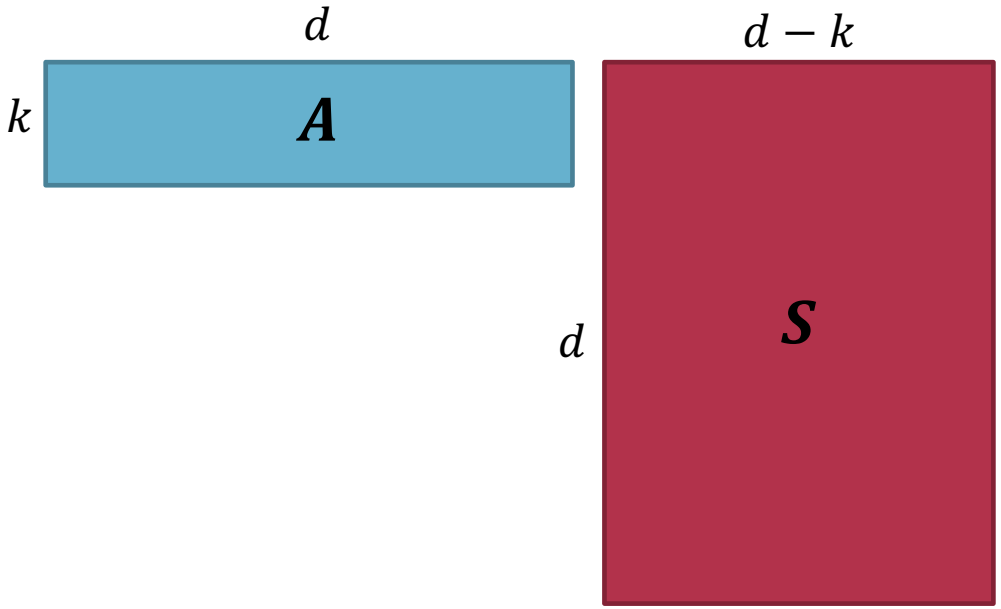
❖ $d - k$ = Rank(T) over $R_q < d$

1. Choose short $S \in R_q^{d \times (d-k)}$

High-level structure of Trapdoor Lattice

Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!
- Rank(T) over R_q < Rank(T) over R $\Longrightarrow$ we can NOT arbitrarily choose all elements of T



$$\equiv \mathbf{0}\ (\boldsymbol{mod}\ \boldsymbol{q})$$

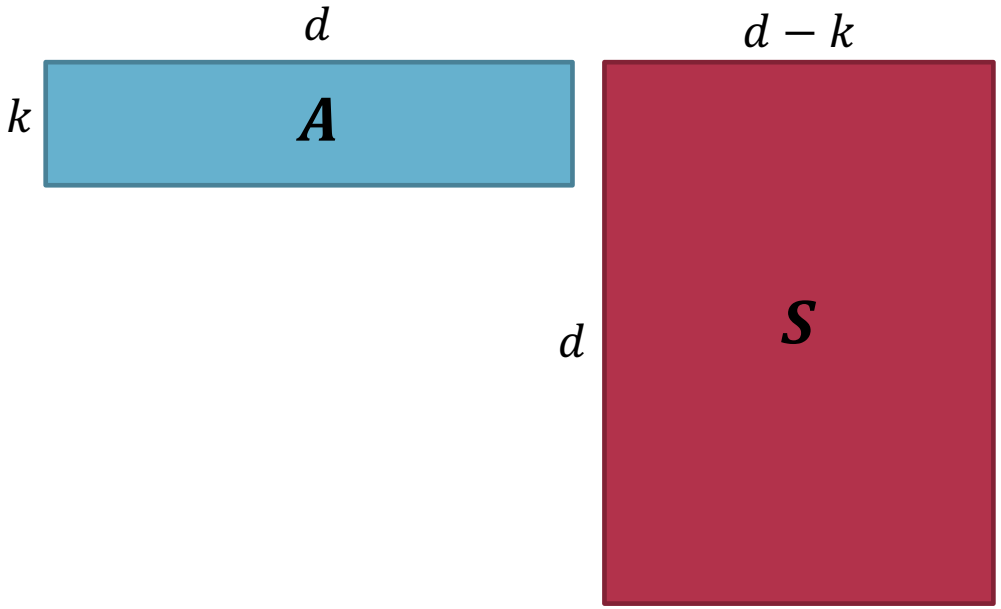
❖ $d - k$ = Rank(T) over $R_q < d$

1. Choose short $S \in R_q^{d\times(d-k)}$
2. Compute its R_q-orthogonal basis A, which does not give S information computationally

High-level structure of Trapdoor Lattice

Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!
- Rank(T) over R_q < Rank(T) over R $\Longrightarrow$ we can NOT arbitrarily choose all elements of T



$$\equiv \mathbf{0}\ (\boldsymbol{mod\ q})$$

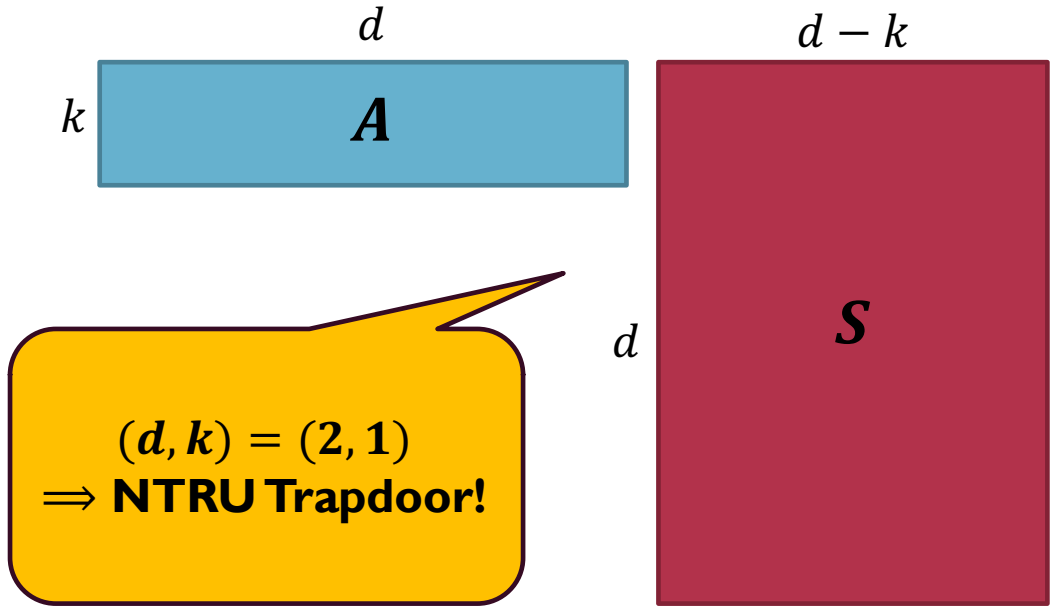
❖ $d - k$ = Rank(T) over $R_q < d$

1. Choose short $S \in R_q^{d\times(d-k)}$
2. Compute its R_q-orthogonal basis A, which does not give S information computationally
3. Generate $T \in R_q^{d\times d}$ s.t. $\Lambda(T) = \Lambda(S||qI_d)$

High-level structure of Trapdoor Lattice

Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!
- Rank(T) over R_q < Rank(T) over R $\Longrightarrow$ we can NOT arbitrarily choose all elements of T



$$\equiv \mathbf{0}\ (\boldsymbol{mod}\ \boldsymbol{q})$$

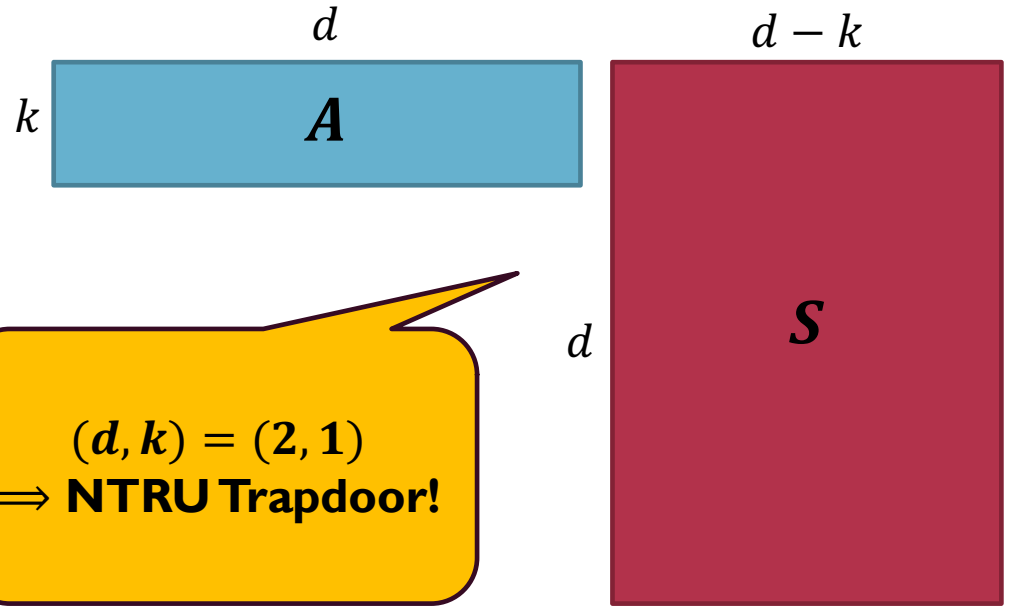
❖ $d - k$ = Rank(T) over $R_q < d$

1. Choose short $S \in R_q^{d\times(d-k)}$
2. Compute its R_q-orthogonal basis A, which does not give S information computationally
3. Generate $T \in R_q^{d\times d}$ s.t. $\Lambda(T) = \Lambda(S||qI_d)$

High-level structure of Trapdoor Lattice

Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!
- Rank(T) over R_q < Rank(T) over R $\Longrightarrow$ we can NOT arbitrarily choose all elements of T



$$\equiv \mathbf{0}\ (\boldsymbol{mod}\ \boldsymbol{q})$$

❖ $d - k$ = Rank(T) over $R_q < d$

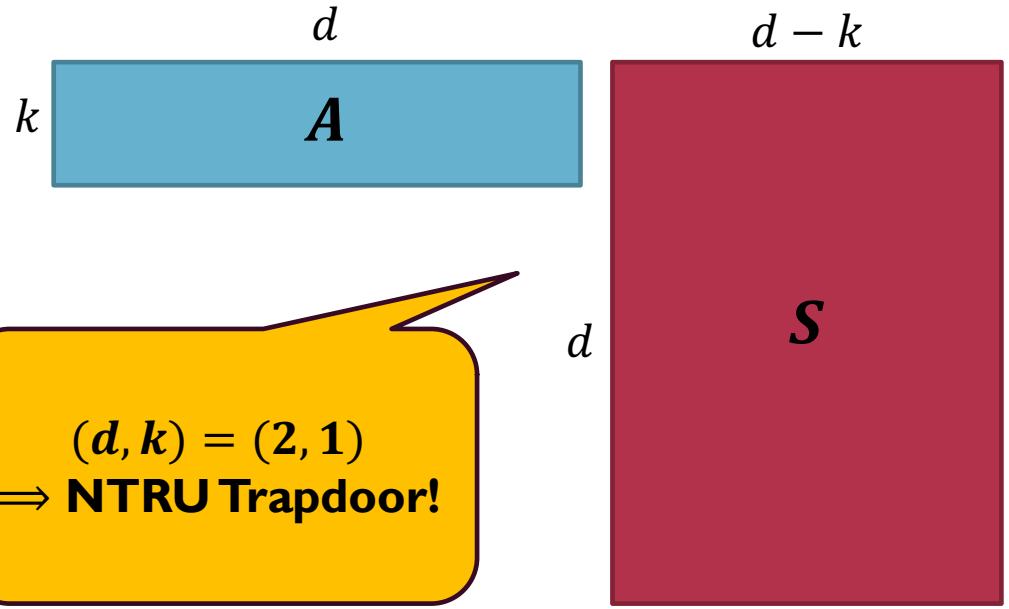
NTRU assumption

1. Choose short $S \in R_q^{d\times(d-k)}$
2. Compute its R_q-orthogonal basis A, which does not give S information computationally
3. Generate $T \in R_q^{d\times d}$ s.t. $\Lambda(T) = \Lambda(S||qI_d)$

High-level structure of Trapdoor Lattice

Observations

- Generate T from A is too hard! $\Longrightarrow$ generate A from T!
- Rank(T) over R_q < Rank(T) over R $\Longrightarrow$ we can NOT arbitrarily choose all elements of T



$\equiv \mathbf{0}\ (\boldsymbol{mod}\ \boldsymbol{q})$

$(\boldsymbol{d}, \boldsymbol{k}) = (\mathbf{2}, \mathbf{1})$
$\Longrightarrow$ **NTRU Trapdoor!**

❖ $d - k$ = Rank(T) over $R_q < d$

NTRU assumption

1. Choose short $S \in R_q^{d\times(d-k)}$
2. Compute its R_q-orthogonal basis A, which does not give S information computationally
3. Generate $T \in R_q^{d\times d}$ s.t. $\Lambda(T) = \Lambda(S||qI_d)$

$\boldsymbol{fG - Fg = q}$

Our Idea

Our Idea – Generalization

Consider the case $(d, k) = (3, 1)$

Our Idea – Generalization

Consider the case $(d, k) = (3, 1)$

Can be Generalized to the case $k = 1$

Our Idea – Generalization

Consider the case $(d, k) = (3, 1)$

Can be Generalized to the case $k = 1$

A_1	A_2	A_3

f_1	f_2
g_1	g_2
h_1	h_2

$$\equiv 0 \ (mod\ q)$$

Our Idea – Generalization

Consider the case $(d, k) = (3, 1)$

Can be Generalized to the case $k = 1$

A_1	A_2	A_3

f_1	f_2
g_1	g_2
h_1	h_2

$\equiv \mathbf{0} \ (\boldsymbol{mod}\ \boldsymbol{q})$

1. Generate "short" (f_i, g_i, h_i) for $i = 1, 2$

Our Idea – Generalization

Consider the case $(d, k) = (3, 1)$

Can be Generalized to the case $k = 1$

A_1	A_2	A_3

f_1	f_2
g_1	g_2
h_1	h_2

$$\equiv \mathbf{0}\ (\boldsymbol{mod}\ \boldsymbol{q})$$

1. Generate "short" (f_i, g_i, h_i) for $i = 1, 2$
2. Set $(A_1, A_2, A_3) = r \cdot (g_1 h_2 - g_2 h_1, h_1 f_2 - h_2 f_1, f_1 g_2 - f_2 g_1)$ for Random $r \in R_q$

Our Idea – Generalization

Consider the case $(d, k) = (3, 1)$

Can be Generalized to the case $k = 1$

A_1	A_2	A_3

f_1	f_2
g_1	g_2
h_1	h_2

$\equiv \mathbf{0}\ (\boldsymbol{mod}\ \boldsymbol{q})$

1. Generate "short" (f_i, g_i, h_i) for $i = 1, 2$
2. Set $(A_1, A_2, A_3) = r \cdot (g_1 h_2 - g_2 h_1, h_1 f_2 - h_2 f_1, f_1 g_2 - f_2 g_1)$ for Random $r \in R_q$
3. Compute "short" (F, G, H) satisfying $\det \begin{pmatrix} f_1 & f_2 & F \\ g_1 & g_2 & G \\ h_1 & h_2 & H \end{pmatrix} = q$

Our Idea – Generalization

Consider the case $(d, k) = (3, 1)$

Can be Generalized to the case $k = 1$

A_1	A_2	A_3

f_1	f_2
g_1	g_2
h_1	h_2

$$\equiv \mathbf{0}\ (\boldsymbol{mod}\ \boldsymbol{q})$$

Multi-instance NTRU assumption

1. Generate "short" (f_i, g_i, h_i) for $i = 1, 2$
2. Set $(A_1, A_2, A_3) = r \cdot (g_1 h_2 - g_2 h_1, h_1 f_2 - h_2 f_1, f_1 g_2 - f_2 g_1)$ for Random $r \in R_q$
3. Compute "short" (F, G, H) satisfying $\det \begin{pmatrix} f_1 & f_2 & F \\ g_1 & g_2 & G \\ h_1 & h_2 & H \end{pmatrix} = q$

Our Idea – Generalization

How to compute "short" (F, G, H) **satisfying** $det\begin{pmatrix} f_1 & f_2 & F \\ g_1 & g_2 & G \\ h_1 & h_2 & H \end{pmatrix} = q$ **?**

Our Idea – Generalization

How to compute "short" (F, G, H) **satisfying** $\boldsymbol{det}\begin{pmatrix} \boldsymbol{f_1} & \boldsymbol{f_2} & \boldsymbol{F} \\ \boldsymbol{g_1} & \boldsymbol{g_2} & \boldsymbol{G} \\ \boldsymbol{h_1} & \boldsymbol{h_2} & \boldsymbol{H} \end{pmatrix} = \boldsymbol{q}$ **?**

- Imitate the NTRU trapdoor computing "short" (F, G) s.t. $fG - Fg = det\begin{pmatrix} f & F \\ g & G \end{pmatrix} = q$

Our Idea – Generalization

How to compute "short" (F, G, H) satisfying $\boldsymbol{det}\begin{pmatrix} \boldsymbol{f_1} & \boldsymbol{f_2} & \boldsymbol{F} \\ \boldsymbol{g_1} & \boldsymbol{g_2} & \boldsymbol{G} \\ \boldsymbol{h_1} & \boldsymbol{h_2} & \boldsymbol{H} \end{pmatrix} = \boldsymbol{q}$?

- Imitate the NTRU trapdoor computing "short" (F, G) s.t. $fG - Fg = det\begin{pmatrix} f & F \\ g & G \end{pmatrix} = q$
  - Assume that $res(f) = \prod f(x^{2i+1}) \in Z$ and $res(g)$ are coprime

Our Idea – Generalization

How to compute "short" (F, G, H) satisfying $\boldsymbol{det}\begin{pmatrix} \boldsymbol{f_1} & \boldsymbol{f_2} & \boldsymbol{F} \\ \boldsymbol{g_1} & \boldsymbol{g_2} & \boldsymbol{G} \\ \boldsymbol{h_1} & \boldsymbol{h_2} & \boldsymbol{H} \end{pmatrix} = \boldsymbol{q}$?

- Imitate the NTRU trapdoor computing "short" (F, G) s.t. $fG - Fg = det\begin{pmatrix} f & F \\ g & G \end{pmatrix} = q$
  - Assume that $res(f) = \prod f(x^{2i+1}) \in Z$ and $res(g)$ are coprime
  - $\alpha \cdot res(f) - \beta \cdot res(g) = 1 \Longrightarrow \left(q\alpha \cdot \prod_{i \neq 0} f\left(x^{2i+1}\right)\right) \cdot f(x) - \left(q\beta \cdot \prod_{i \neq 0} g\left(x^{2i+1}\right)\right) \cdot g(x) = q$

Our Idea – Generalization

How to compute "short" (F, G, H) satisfying $\boldsymbol{det}\begin{pmatrix} \boldsymbol{f_1} & \boldsymbol{f_2} & \boldsymbol{F} \\ \boldsymbol{g_1} & \boldsymbol{g_2} & \boldsymbol{G} \\ \boldsymbol{h_1} & \boldsymbol{h_2} & \boldsymbol{H} \end{pmatrix} = \boldsymbol{q}$?

- Imitate the NTRU trapdoor computing "short" (F, G) s.t. $fG - Fg = det\begin{pmatrix} f & F \\ g & G \end{pmatrix} = q$
  - Assume that $res(f) = \prod f(x^{2i+1}) \in Z$ and $res(g)$ are coprime
  - $\alpha \cdot res(f) - \beta \cdot res(g) = 1 \Rightarrow \underbrace{\left(q\alpha \cdot \prod_{i \neq 0} f\left(x^{2i+1}\right)\right)}_{:= G_0} \cdot f(x) - \underbrace{\left(q\beta \cdot \prod_{i \neq 0} g\left(x^{2i+1}\right)\right)}_{:= F_0} \cdot g(x) = q$

Our Idea – Generalization

How to compute "short" (F, G, H) satisfying $\boldsymbol{det}\begin{pmatrix} \boldsymbol{f_1} & \boldsymbol{f_2} & \boldsymbol{F} \\ \boldsymbol{g_1} & \boldsymbol{g_2} & \boldsymbol{G} \\ \boldsymbol{h_1} & \boldsymbol{h_2} & \boldsymbol{H} \end{pmatrix} = \boldsymbol{q}$?

- Imitate the NTRU trapdoor computing "short" (F, G) s.t. $fG - Fg = det\begin{pmatrix} f & F \\ g & G \end{pmatrix} = q$
  - Assume that $res(f) = \prod f(x^{2i+1}) \in Z$ and $res(g)$ are coprime
  - $\alpha \cdot res(f) - \beta \cdot res(g) = 1 \Longrightarrow \underbrace{\left(q\alpha \cdot \prod_{i\neq 0} f\left(x^{2i+1}\right)\right)}_{:= G_0} \cdot f(x) - \underbrace{\left(q\beta \cdot \prod_{i\neq 0} g\left(x^{2i+1}\right)\right)}_{:= F_0} \cdot g(x) = q$
  - Reduce (F_0, G_0) with a "short" vector (f, g), and output (F, G)

Our Idea – Generalization

How to compute "short" (F, G, H) satisfying $det\begin{pmatrix} f_1 & f_2 & F \\ g_1 & g_2 & G \\ h_1 & h_2 & H \end{pmatrix} = q$?

- Our Case

Our Idea – Generalization

How to compute "short" (F, G, H) satisfying $det\begin{pmatrix} f_1 & f_2 & F \\ g_1 & g_2 & G \\ h_1 & h_2 & H \end{pmatrix} = q$?

- Our Case
  - Assume $\gcd(res(f_1 g_2 - f_2 g_1), res(g_1 h_2 - g_2 h_1), res(h_1 f_2 - h_2 f_1)) = 1$

Our Idea – Generalization

How to compute "short" (F, G, H) satisfying $\boldsymbol{det}\begin{pmatrix} \boldsymbol{f_1} & \boldsymbol{f_2} & \boldsymbol{F} \\ \boldsymbol{g_1} & \boldsymbol{g_2} & \boldsymbol{G} \\ \boldsymbol{h_1} & \boldsymbol{h_2} & \boldsymbol{H} \end{pmatrix} = \boldsymbol{q}$?

- Our Case
  - Assume $\gcd(res(f_1g_2 - f_2g_1), res(g_1h_2 - g_2h_1), res(h_1f_2 - h_2f_1)) = 1$
  - Obtain (F_0, G_0, H_0) satisfying $F_0 \cdot (g_1h_2 - g_2h_1) + G_0 \cdot (h_1f_2 - h_2f_1) + H_0 \cdot (f_1g_2 - f_2g_1) = q$

Our Idea – Generalization

How to compute "short" (F, G, H) satisfying $\boldsymbol{det}\begin{pmatrix} \boldsymbol{f_1} & \boldsymbol{f_2} & \boldsymbol{F} \\ \boldsymbol{g_1} & \boldsymbol{g_2} & \boldsymbol{G} \\ \boldsymbol{h_1} & \boldsymbol{h_2} & \boldsymbol{H} \end{pmatrix} = \boldsymbol{q}$?

- Our Case
  - Assume $\gcd(res(f_1g_2 - f_2g_1), res(g_1h_2 - g_2h_1), res(h_1f_2 - h_2f_1)) = 1$
  - Obtain (F_0, G_0, H_0) satisfying $F_0 \cdot (g_1h_2 - g_2h_1) + G_0 \cdot (h_1f_2 - h_2f_1) + H_0 \cdot (f_1g_2 - f_2g_1) = q$
  - Reduce (F_0, G_0, H_0) with "short" vectors (f_1, g_1, h_1) and (f_2, g_2, h_2)

Ongoing Works and Expectation

Ongoing Works and Expectation

- Analysis
  - Cryptanalysis on Multi-instance NTRU
  - Analysis on Signature size

- Expectation
  - Better Flexibility on Parameters than Falcon
  - Generalization of specific NTRU Trapdoor
  - What else..?

thank you!