耐量子計算機暗号の解説
～符号暗号を中心として～

草川恵太（NTT社会情報研究所）
2021/12/08 SITA2021

PQC

Quantum Computers

Quantum Computers:

Google: 54 qubits (2019)

IBM Q53: 53 qubits (2019)

USTC: 76 qubits (2020)

QC solves factoring/DL [Sho94]

→  may be quantum

cf. https://en.wikipedia.org/wiki/List_of_quantum_processors
[Sho94] Shor (FOCS 1994)

Countermeasures

Quantum Computers:

Google: 54 qubits (2019)

IBM Q53: 53 qubits (2019)

USTC: 76 qubits (2020)

QC solves factoring/DL [Sho94]

→  may be quantum

Countermeasures:

1. Looooooooooong RSA/DL
2. Quantum cryptography
   QKD etc.
3. Post-quantum cryptography
   (PQC)

cf. https://en.wikipedia.org/wiki/List_of_quantum_processors
[Sho94] Shor (FOCS 1994)

1. Looooooooooong RSA/DL

NTT

Why PQC?

QC solves factoring/DL [Sho94]

→ 👹 may be quantum

… Is RSA/DL broken?

PQ RSA [BFHLV17]:

pqrsa30: 2^{30}-byte keys using 1024-bit primes

2^{110} T-gates with 2^{34} qubits

Pros: Run on classical machines

Cons: Too slow※

- Keygen: 2.3 days
- Dec: 2.1 hours
- Enc: 10.1 minutes

[BFHLV17] Bernstein, Fried, Heninger, Lou, Valenta (NIST PQC Round 1)
[BHLV17] Bernstein, Heninger, Lou, Valenta (PQCRYPTO 2017)
※ 1core 3GHz Intel Skylake

2. Quantum cryptography

Why PQC?

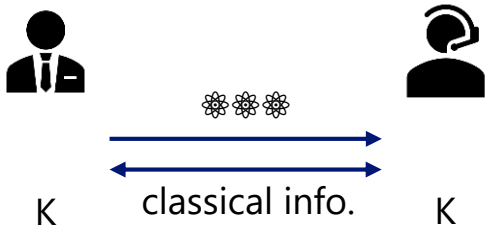
QC solves factoring/DL [Sho94]

→ 👹 may be quantum

… Use quantum machinery

QKD [BB84,…]:

Share random keys using quantum bits



Pros: (Physically) secure

Cons: Require special machines

[BB84] Bennett, Brassard (International Conference on Computers, Systems & Signal Processing 1984)

3. Post-quantum cryptography (PQC)

NTT

Why PQC?

QC solves factoring/DL [Sho94]

→  may be quantum

… Hard problems for QC

PQC:

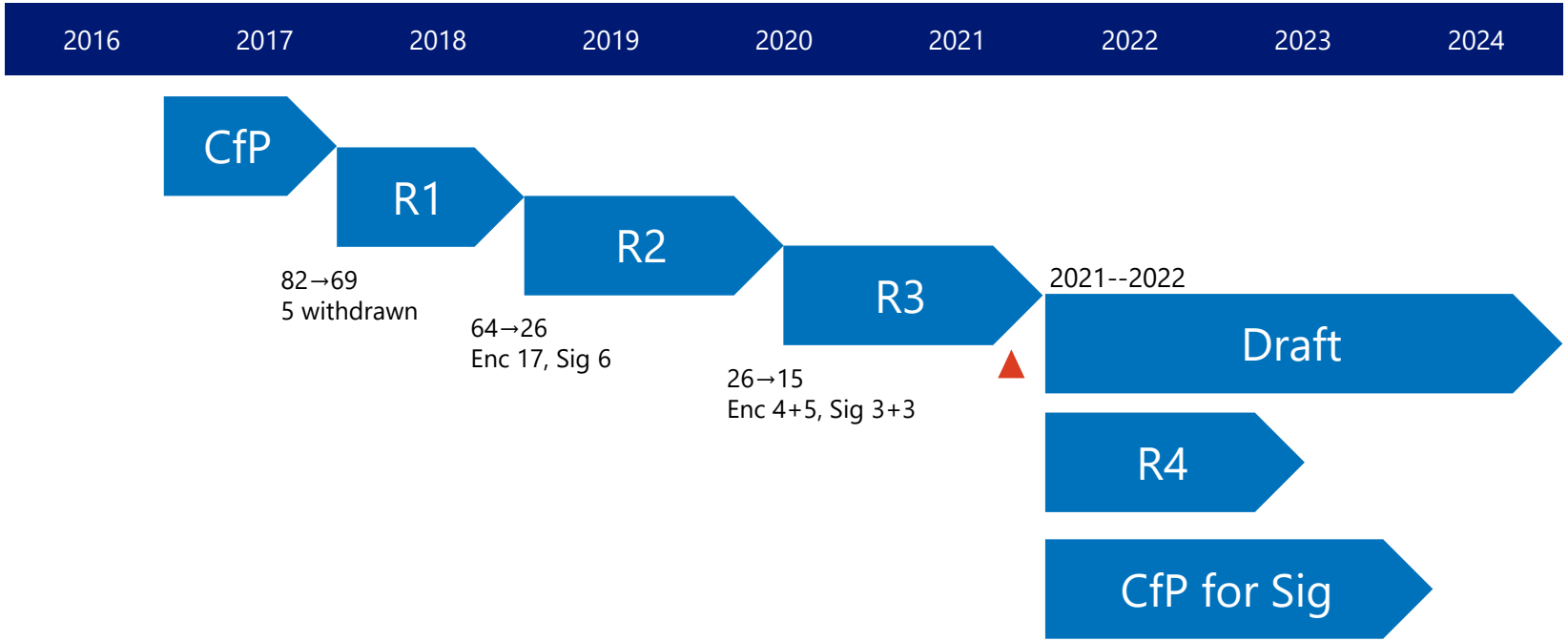
Use classical cryptography based on computational problems resilient to quantum attacks

Pros: Run on classical machines

Cons: Computational assumptions

NIST PQC

NIST PQC Standardization Timeline



https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline

NIST PQC Round 1

69 candidates are accepted from 82 submissions

BIG QUAKE, BIKE, CFPKM, Classic McEliece, Compact LWE, CRYSTALS-Dilithium, CRYSTALS-Kyber, DAGS, Ding Key Exchange, DME, DRS, DualModeMS, Edon-K, EMBLEM and R.EMBLEM, Falcon, FrodoKEM, GeMSS, Giophantus, Gravity-SPHINCS, GuessAgain, Gui, Hila5, HiMQ-3, HK17, HQC, KCL, KINDI, LAC, LAKE, LEDAkem, LEDApkc, Lepton, Lima, Lizard, LOCKER, LOTUS, LUOV, McNie, Mersenne-756839, MQDSS, NewHope, NTRUEncrypt, pqNTRUsign, NTRU-HRSS-KEM, NTRU Prime, NTS-KEM, Odd Manhattan, Ouroboros-R, Picnic, Post-Quantum RSA-Encryption, Post-Quantum RSA-Signature, pqsigRM, QC-MDPC KEM, qTESLA, RaCoSS, Rainbow, Ramstake, RankSign, RLCE-KEM, Round2, RQC, RVB, SABER, SIKE, SPHINCS+, SRTPI, Three Bears, Titanium, WalnutDSA

NIST PQC Round 1 (2017-12-21—)

Enc 49:

Code 17: BIG QUAKE, BIKE, Classic McEliece, DAGS, Edon-K, HQC, LAKE, LEDAkem, LEDApkc, Lepton, LOCKER, McNie, NTS-KEM, Ouroboros-R, QC-MDPC KEM, RLCE-KEM, RQC
Lattice 22: Compact LWE, CRYSTALS-Kyber, Ding Key Exchange, EMBLEM and R.EMBLEM, FrodoKEM, Giophantus, Hila5, KCL, KINDI, LAC, Lima, Lizard, LOTUS, NewHope, NTRUEncrypt, NTRU-HRSS-KEM, NTRU Prime, Odd Manhattan, Round2, SABER, Titanium, Three Bears
MQ 3: CFPKM, DME, SRTPI
Isogeny 1: SIKE
Mersenne 2: Mersenne-756839, Ramstake
Others 4: Post-Quantum RSA-Encryption, RVB, GuessAgain, HK17

Sig 22:

Code 3: pqsigRM, RaCoSS, RankSign
Lattice 5: CRYSTALS-Dilithium, DRS, Falcon, pqNTRUsign, qTESLA
MQ 8: DME, DualModeMS, GeMSS, Gui, HiMQ-3, LUOV, MQDSS, Rainbow
Sym/Hash 3: Gravity-SPHINCS, Picnic, SPHINCS+
Others 2: Post-Quantum RSA-Signature, WalnutDSA

NIST PQC Round 2 (2019-01-30—)

Enc 49→17:

Code 17→7: BIG BIKE, Classic McEliece, HQC, LEDAcrypt(=LEDAkem+LEDApkc), NTS-KEM, ROLLO(=LAKE+LOCKER+Ouroboros-R), RQC
Lattice 22→9: CRYSTALS-Kyber, FrodoKEM, LAC, NewHope, NTRU(=NTRUEncrypt+NTRU-HRSS-KEM), NTRU Prime, Round5(=HIla5+Round2), SABER, Three Bears
MQ 3→0:
Isogeny 1→1: SIKE
Mersenne 2→0:
Others 4→0:

Sig 22→9:

Code 3→0:
Lattice 5→3: CRYSTALS-Dilithium, Falcon, qTESLA
MQ 8→4: GeMSS, LUOV, MQDSS, Rainbow
Sym/Hash 3→2: Picnic, SPHINCS+
Others 2→0:

NIST PQC Round 3 (2020-07-22—)

4 Finalists: Enc

Classic McEliece* (code)
CRYSTALS-Kyber (lattice)
NTRU (lattice)
SABER (lattice)

3 Finalists: Sig

CRYSTALS-Dilithium (lattice)
Falcon (lattice)
Rainbow (MQ)

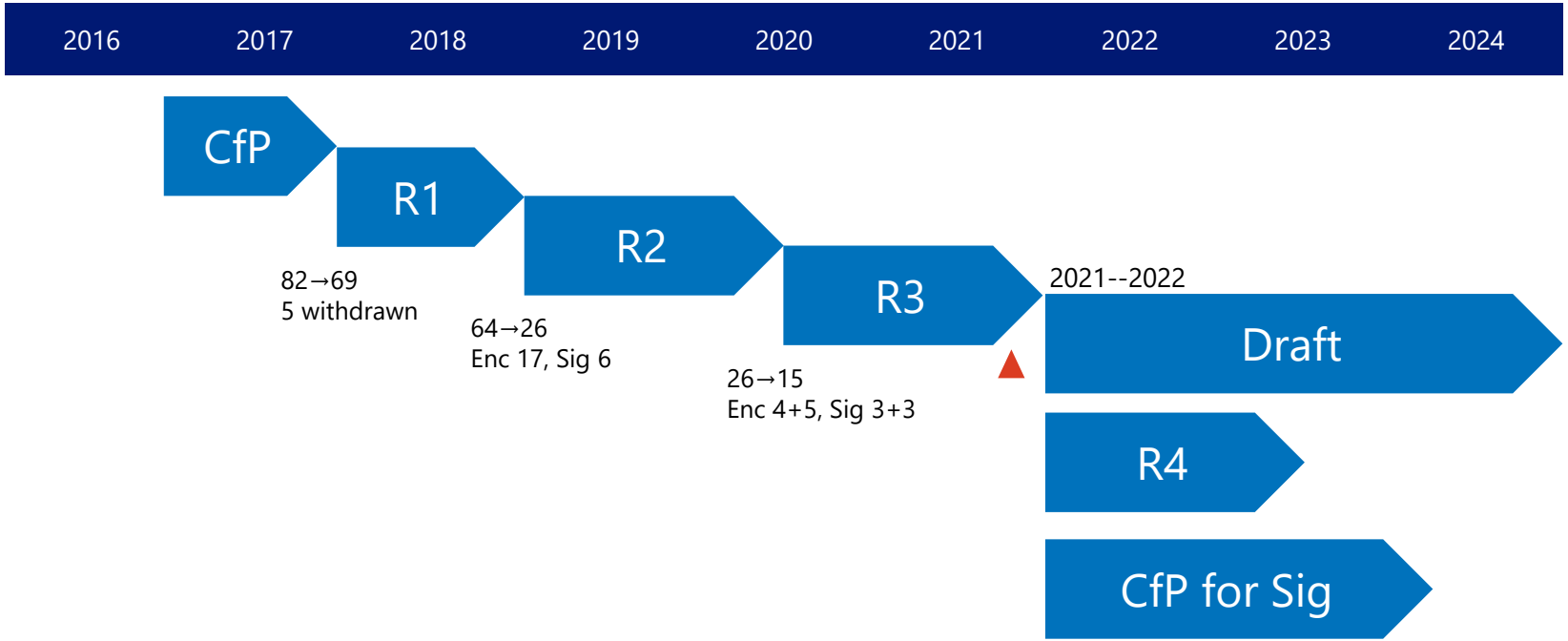
5 Alternates: Enc

BIKE (code)
FrodoKEM (lattice)
HQC (code)
NTRU Prime (lattice)
SIKE (isogeny)

3 Alternates: Sig

GeMSS (MQ)
Picnic (ZK)
SPHINCS+ (Hash)

NIST PQC Standardization Timeline



https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline

NIST PQC Round 2 (2019-01-30—)

Enc 49→17:

Code 17→7: BIG BIKE, Classic McEliece, HQC, LEDAcrypt(=LEDAkem+LEDApkc), NTS-KEM, ROLLO(=LAKE+LOCKER+Ouroboros-R), RQC
Lattice 22→9: CRYSTALS-Kyber, FrodoKEM, LAC, NewHope, NTRU(=NTRUEncrypt+NTRU-HRSS-KEM), NTRU Prime, Round5(=HIla5+Round2), SABER, Three Bears
MQ 3→0:
Isogeny 1→1: SIKE
Mersenne 2→0:
Others 4→0:

Sig 22→9:

Code 3→0:
Lattice 5→3: CRYSTALS-Dilithium, Falcon, qTESLA
MQ 8→4: GeMSS, LUOV, MQDSS, Rainbow
Sym/Hash 3→2: Picnic, SPHINCS+
Others 2→0:

Category of PQC

	Enc	Sig
Lattice	Good	Good
Code	Good	???
MQ	???	Good
Isogeny	Good	(may good)
SKE/Hash	Infeasible	Good

Lattice-based cryptography

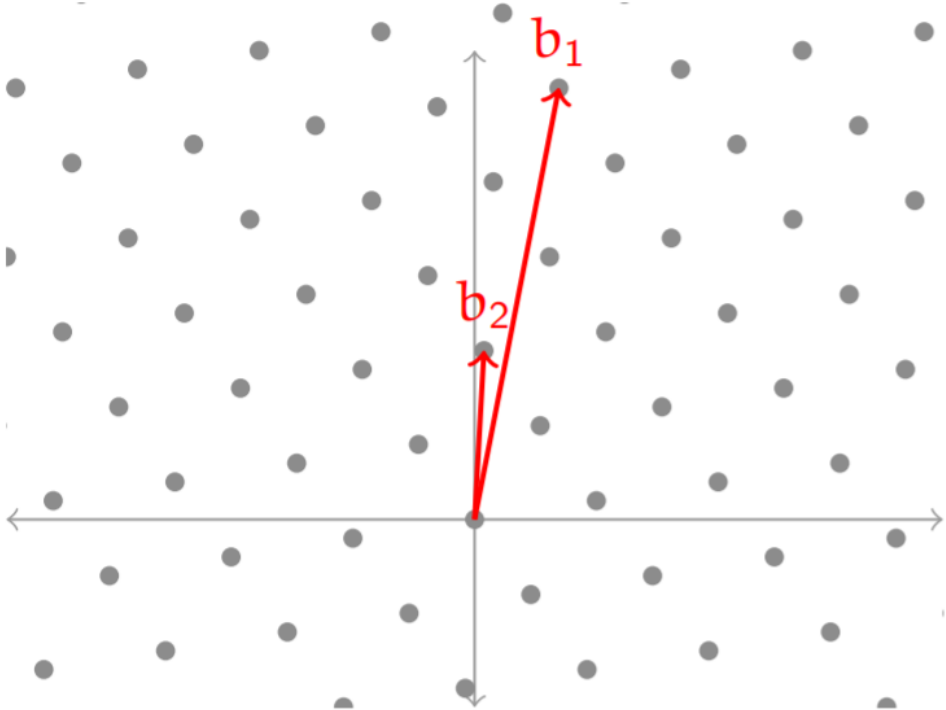
Lattice

- $B = [b_1, \ldots, b_n] \in \mathbb{Z}^n$
- Lattice spanned by B
  $L(B) \coloneqq \{\sum_i \alpha_i b_i \mid \alpha_i \in \mathbb{Z}\}$

Lattice-based cryptography

Cryptography based on lattice problems

(Use Learning With Errors (LWE))



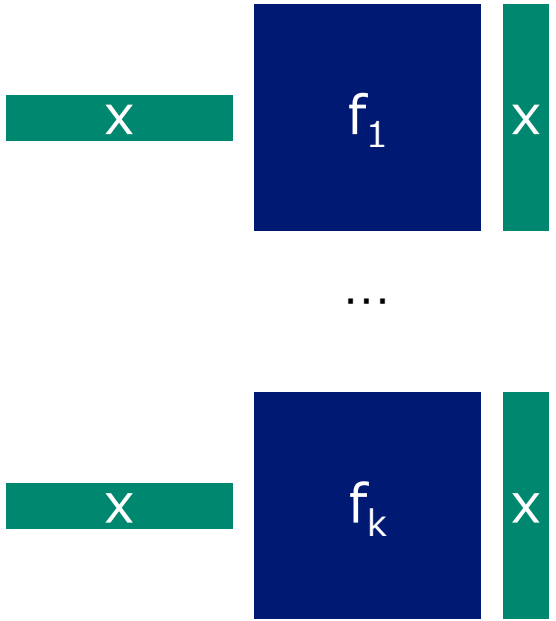
MQ-based cryptography

MQ: multivariate quadratic functions

- $F=(f_1,\ldots,f_k)$ with f_i in $F_q[x_1,\ldots,x_n]$

MQ-based cryptography

Cryptography based on MQ problems



Isogeny-based cryptography

Isogeny: Morphism between elliptic curves E, E′

- φ: E → E′ is isogeny

Isogeny-based cryptography

Cryptography using isogeny

Code-based cryptography

Enc:

- Classic McEliece* (finalist)
- BIKE (alternate candidate)
- HQC (alternate candidate)

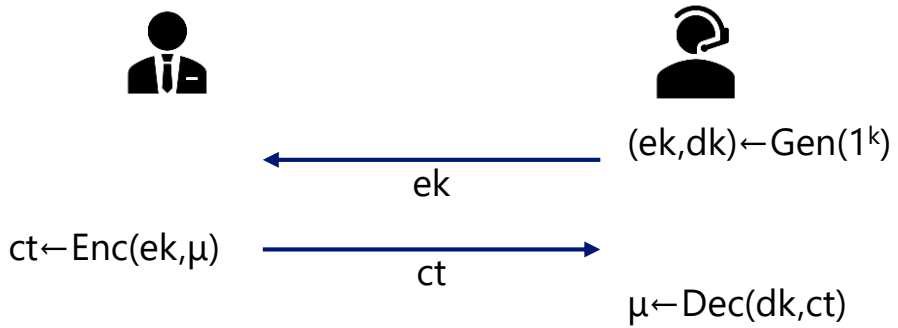
Sig:

- CFS (--)

Code-based PKEs

PKE – Public-Key Encryption

- $Gen(1^k) \rightarrow (ek,dk)$
- $Enc(ek,\mu) \rightarrow ct$
- $Dec(dk,ct) \rightarrow \mu/\perp$

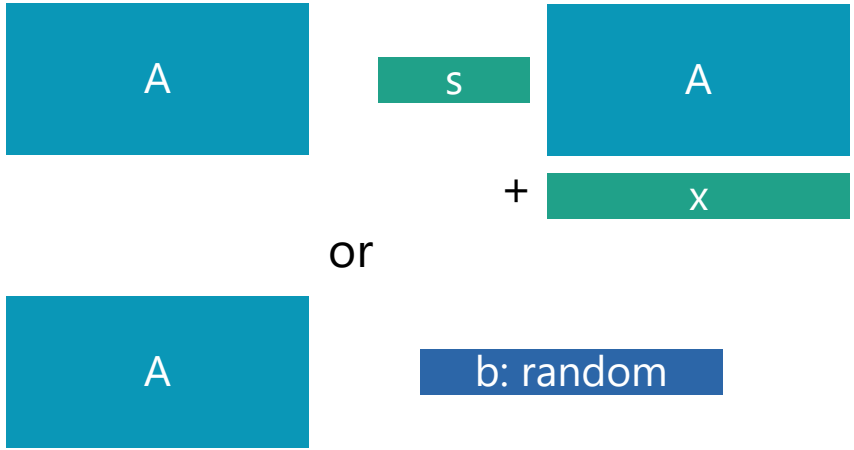


- ek = encryption key
- dk = decryption key
- ct = ciphertext

Learning Parity with Noise (LPN)

Distinguish

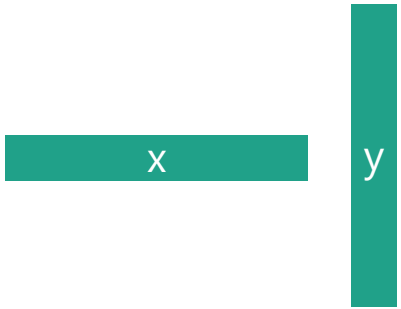
- (A,sA+x), where $A \leftarrow \mathbb{Z}_2^{k \times n}$, $s \leftarrow \mathbb{Z}_2^k$, $x \leftarrow \chi^n$
- (A,b), where $A \leftarrow \mathbb{Z}_2^{k \times n}$, $b \leftarrow \mathbb{Z}_2^n$
- $\chi = \chi_p$: Bernoulli distribution
  - return 0 with prob. 1-p
  - return 1 with prob p



- For a random code, a random codeword with error looks like random!
- We can replace $s \leftarrow \mathbb{Z}_2^k$ with $s \leftarrow \chi^k$

Note on Bernoulli distribution

- $\chi=\chi_p$: Bernoulli distribution
  - return 0 with prob. 1-p
  - return 1 with prob p
- Consider x, y ← χ_p^n

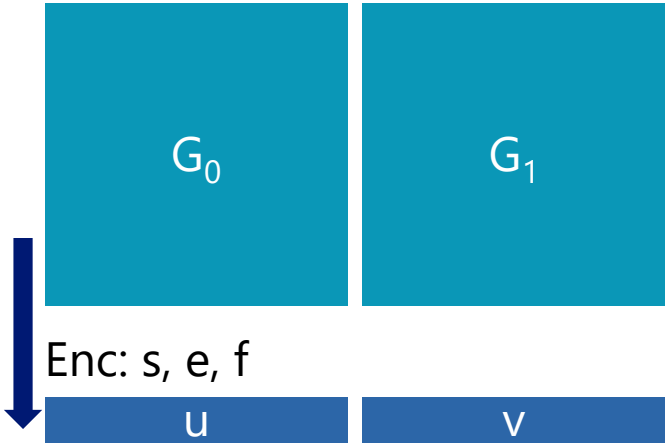


Lemma: If $p = n^{-1/2-c/2}$ for small c,

$$\Pr[\langle x, y\rangle = 1] \leq 2n^{-c}.$$

- If there is no index i with $x_i=y_i=1$, then $\langle x, y\rangle = 0$
- $\Pr[\langle x, y\rangle = 0] \geq \Pr[\forall i: \neg(x_i = y_i = 1)]$ $= (1-p^2)^n$
- $\Pr[\langle x, y\rangle = 1] \leq 1-(1-p^2)^n$ $\leq 1-(1-2n\,p^2) = 2n^{-c}$

HQC.PKE simplified



- G_0: random
- $G_1 = G_0X+Y$, where $X,Y \leftarrow \chi^{k \times k}$
- $u = sG_0+e$, where $s,e \leftarrow \chi^k$
- $v = sG_1+f+enc(m)$, where $f \leftarrow \chi^k$

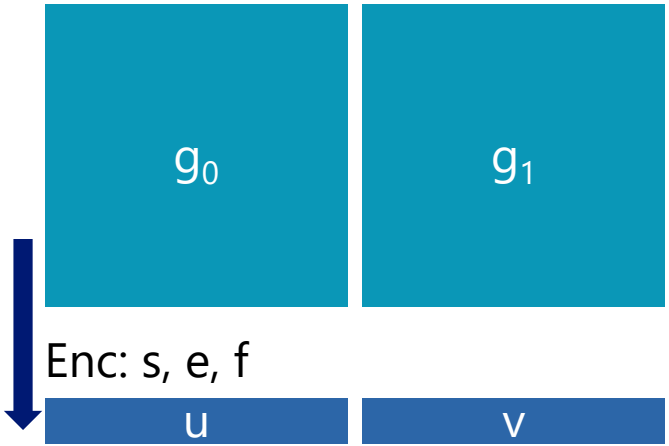
- $ek=(G_0,G_1)$, $dk=X$, $ct=(u,v)$

$v - uX$
$=sG_1+f+enc(m) - (sG_0+e)X$
$=s(G_0X+Y)+f+enc(m) - (sG_0+e)X$
$=enc(m) +sY+f-eX$
$\rightsquigarrow$ m by decode since $HW(err) \leq 2k^{1-c}$

Proof idea:

1. LPN: $(G_0,G_1) \approx_c G'=(G_0,random)$
2. LPN: $(G',sG'+(e,f)) \approx_c (G',random)$

補足: HQC.PKE (Quasi-Cyclic)



- g_0: random in $\mathbb{Z}_2[\alpha]/(\alpha^k - 1)$
- $g_1 = g_0x+y$, where $x,y \leftarrow \chi^k$
- $u = sg_0+e$, where $s,e \leftarrow \chi^k$
- $v = sg_1+f+enc(m)$, where $f \leftarrow \chi^k$

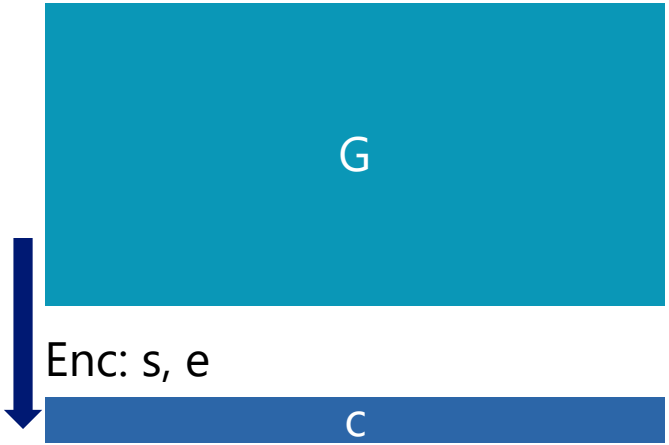
- $ek=(g_0,g_1)$, $dk=x$, $ct=(u,v)$

$v - ux$
$=sg_1+f+enc(m) - (sg_0+e)x$
$=s(g_0x+y)+f+enc(m) - (sg_0+e)x$
$=enc(m) +sy+f-ex$
$\rightsquigarrow$ m by decode since HW(err) is small

Proof idea:

1. RingLPN: $(g_0,g_1) \approx_c G'=(g_0,random)$
2. RingLPN: $(G',sG'+(x,y)) \approx_c (G',random)$

McEliece PKE [McE78]



- G^+: random bin. Goppa code
- $G=SG^+P$, $S \leftarrow GL(F_2,k)$, $P \leftarrow Perm(n)$
- $c=sG+e$, where $e \leftarrow \chi^n$

- ek=G, dk=(S,G^+,P), ct=c

1. $c\,P^{-1} = sSG^+ + eP^{-1} = sSG^+ + e'$
2. $\rightsquigarrow sS$ by decode of G^+
3. return s and e

Proof idea:

1. KI: $G \approx_c G'$=random
2. LPN: $(G',sG'+e) \approx_c (G',\text{random})$

[McE78] McEliece (1978)

Niederreiter PKE [Nie78]

- ek=H, dk=(S,H^+,P), ct=u

1. $S^{-1}u = S^{-1}He = S^{-1}SH^+Pe = H^+Pe$
2. ↝Pe by decode of H^+
3. return e

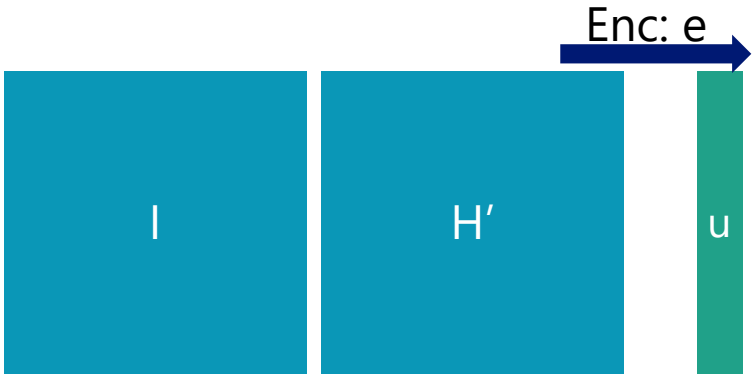
- H^+: par. mat. of a code
- $H=SH^+P$, $S \leftarrow GL(F_2, n-k)$, $P \leftarrow Perm(n)$
- $u=He$, where $e \leftarrow \chi^n$

Security:

McEliece PKE=Niederreiter PKE [LDW94]
Consider a dual of code gen. by ek

[Nie84] Niederreiter (1986)
[LDW94] Li, Deng, Wang (IEEE Trans. on IT 1994)

Classic McEliece (=Niederreiter PKE)



- ek=H’, dk=(S,H^+,P), ct=u

1. $S^{-1}u = S^{-1}He = S^{-1}SH^+Pe = H^+Pe$
2. ⇝Pe by decode of H^+
3. return e

- H^+: par. mat. of a binary Goppa code
- $H=SH^+P$ s.t. H: systematic, i.e., H=[I,H’]
- u=He, where e←{x: HW(x)=t}

BIKE.PKE simplified

- $H_0, H_1 \leftarrow F_2^{n \times n}$: low-weight
- $H = H_1^{-1} H_0$
- $u = [I,H] (x,y) = x + Hy$, where $x,y \leftarrow \chi^n$

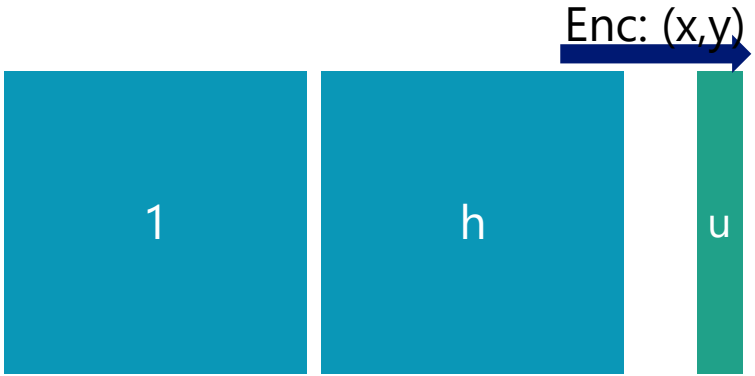
- ek=H, dk=(H_0,H_1), ct=u

1. $H_1 u = H_1(x+Hy) = H_1 x + H_0 y = [H_1,H_0] (x,y)$
2. $\rightsquigarrow(x,y)$ by decode (MDPC)

Proof idea:

1. KI: $H \approx_c H'$ =random
2. LPN: $(H', x+H'y) \approx_c (H', \text{random})$

補足: BIKE.PKE (Quasi-Cyclic)



- ek=h, dk=(h_0,h_1), ct=u

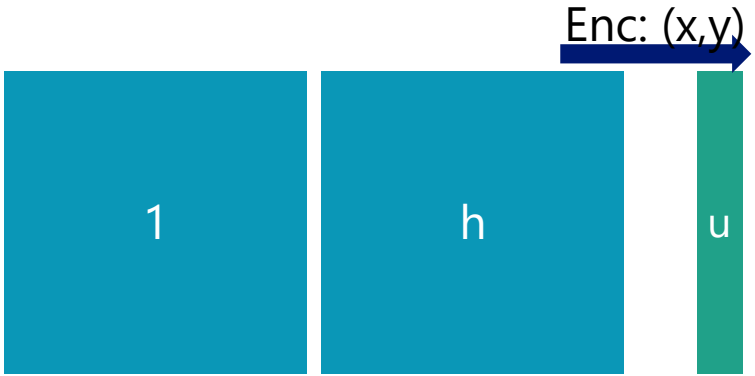
1. $h_1u = h_1(x+hy) = h_1x + h_0y$ $=[h_1,h_0]\ (x,y)$
2. $\rightsquigarrow$(x,y) by decode QC-MDPC

- h_0,h_1: low-weight in $\mathbb{Z}_2[\alpha]/(\alpha^k - 1)$
- h=h_0/h_1
- u=[1,h] (x,y) = x+hy, where x,y$\leftarrow\chi^n$

Proof idea:

1. KI: h $\approx_c$ h′ =random
2. RingLPN: (h′,x+h′y) $\approx_c$ (h′,random)

補足: BIKE.PKE (Quasi-Cyclic)



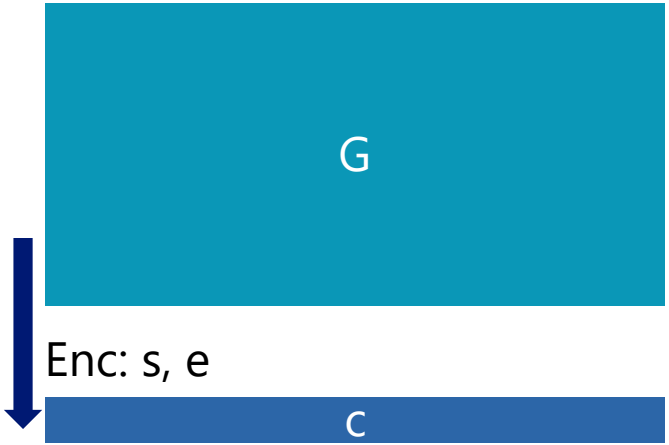
- n: 10,000 – 100,000
- QC-LDPC: $d_c \approx \log(n)$
- QC-MDPC: $d_c \approx \sqrt{n}$
- QC-LDPC is vulnerable [OTD10]

- h_0,h_1: low-weight in $\mathbb{Z}_2[\alpha]/(\alpha^k - 1)$
- $h = h_0/h_1$
- u=[1,h] (x,y) = x+hy, where x,y←χ^n

- Decode of QC-MDPC:
  - Old: Bit-Flipping (DFR=10^{-7})
  - New: Black-Gray-Flip (DFR=2^{-128}?)

[OTD08, 10] Otmani, Tillich, Dallot (https://arxiv.org/abs/0804.0409, Mathematics in Computer Science 2010)

Variant of McEliece/Niederreiter



- G^+: random bin. Goppa code
- $G=SG^+P$, $S \leftarrow GL(F_2,k)$, $P \leftarrow Perm(n)$
- $c=sG+e$, where $\leftarrow \chi^n$

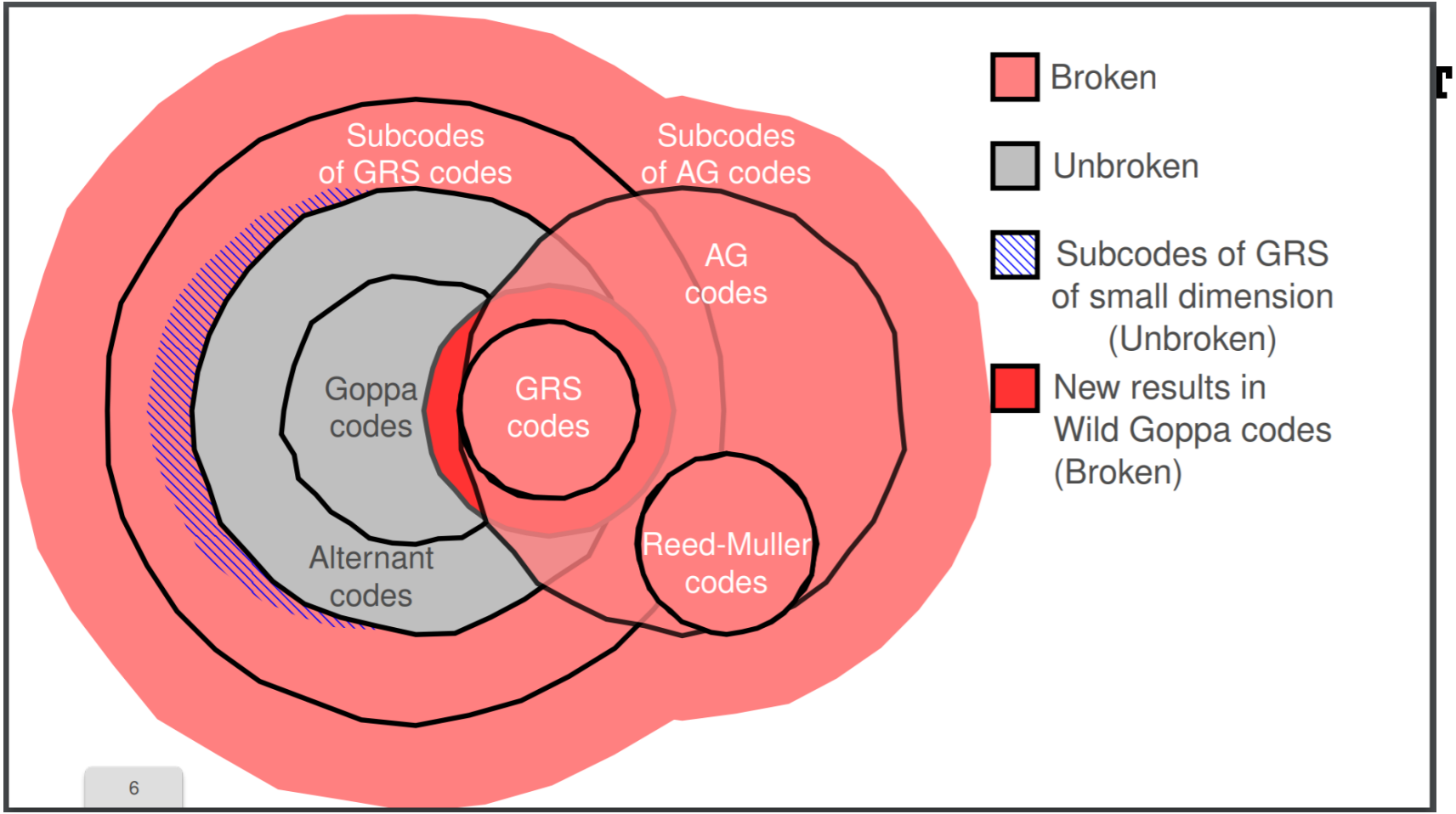
- $ek=G$, $dk=(S,G^+,P)$, $ct=c$

Cons: ek is huge, $O(k^2)$

Idea: Use structured code!

RS, Generalized RS, Quasi-Cyclic-LDPC, Quasi-Dyadic, and so on…

Attacks: Structure is exploited…



Irene Márquez-Corbella: Code-Based Cryptography > 4: Key Attacks > 4.9. Goppa codes still resist
https://www.canal-u.tv/video/inria/4_9_goppa_codes_still_resist.32955
https://streaming-canal-u.fmsh.fr/vod/media/canalu/documents/fuscia/4.9.goppa.codes.still.resist_32955/c015im.w4.s9.v1.pdf

Generic Analysis

- Challenges for code-based problems https://decodingchallenge.org/
- **Syndrome Decoding (SD)**: Finding e from H and He

Hall of Fame

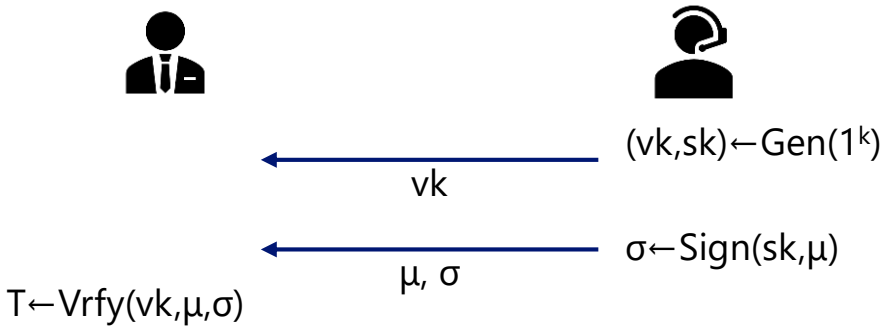
- SD in random: n=530 (2021/10/27 by Narisada et al.)
  - $w=\mathrm{ceil}(1.05d_{GV})$
- SD in CME: n=1284, w=24 (2021/08/16 by Esser et al.)
- SD in QC: n=2918, w=54 (2021/05/31 by Esser et al.)

[Narisada et al.] See https://www.kddi-research.jp/topics/2021/093001.html
[EMZ21] Esser, May, Zweydinger (https://eprint.iacr.org/2021/1634)

Code-based signature

Sig – Digital Signature

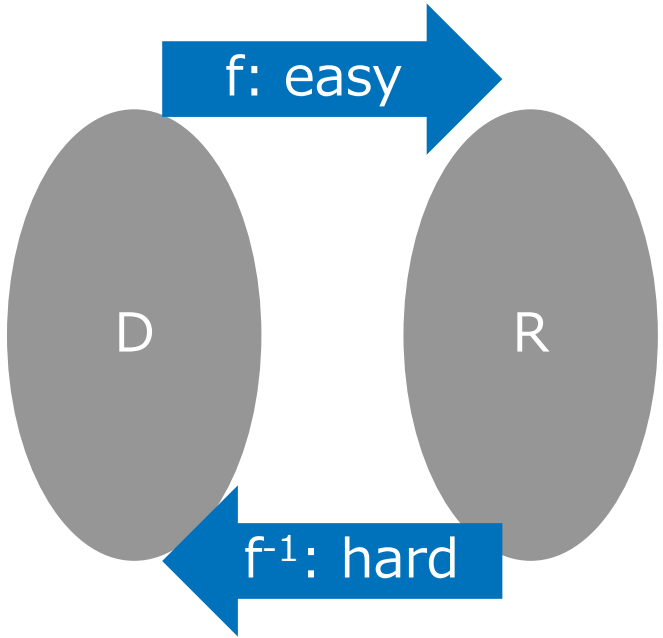
- $\mathrm{Gen}(1^k) \rightarrow (vk, sk)$
- $\mathrm{Sign}(sk, \mu) \rightarrow \sigma$
- $\mathrm{Vrfy}(vk, \mu, \sigma) \rightarrow T / \perp$



- vk = verification key
- sk = signing key
- σ = signature

Signature w/ TDF – Simplified

- Consider f: D → R, hash: $\{0,1\}^* \rightarrow R$
- vk = f, sk = f^{-1}
- Sign(sk,μ) = f^{-1}(hash(μ))
- Vrfy(vk,μ,σ): check if hash(μ) = f(σ)



Niederreiter PKE [Nie78]

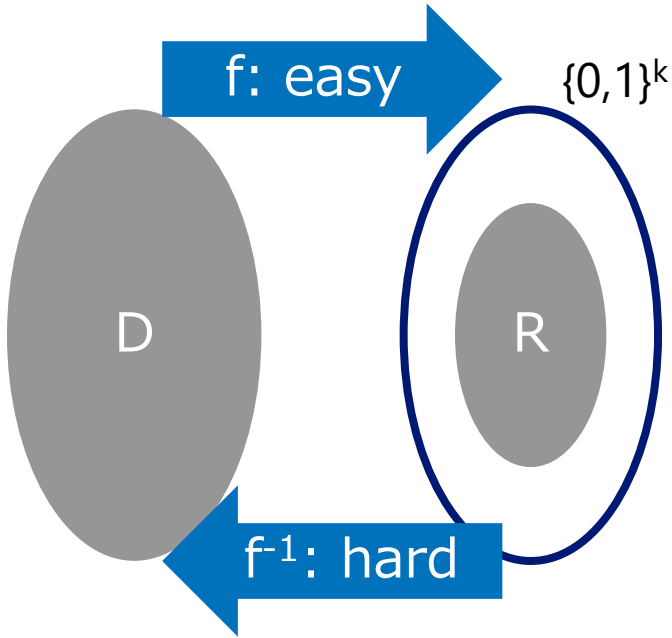
Decryption

1. $S^{-1}u = S^{-1}He = S^{-1}SH^{+}Pe = H^{+}Pe$
2. $\rightsquigarrow Pe$ by decode and return e

- H^{+}: par. mat. of a code
- $H=SH^{+}P$, $S \leftarrow GL(F_2, n-k)$, $P \leftarrow Perm(n)$
- $u=He$, where $e \leftarrow \chi^n$
- $ek=H$, $dk=(S,H^{+},P)$, $ct=u$

Signature w/ TDF small range

- Consider f: D → R, hash: $\{0,1\}^* \rightarrow \{0,1\}^k$
- vk = f, sk = f^{-1}
- Sign(sk,μ):
  - Try i to hash(μ,i) in R
  - If so, compute f^{-1}(hash(μ,i))
- Vrfy(vk,μ,(σ,i)): check if hash(μ,i) = f(σ)



CFS signature [CFS01]

- H^+: par. mat. of a code
- $H=SH^+P$, $S \leftarrow GL(F_2, n-k)$, $P \leftarrow Perm(n)$
- $u=He$, where $e \leftarrow \chi^n$
- $vk=H$, $sk=(S,H^+,P)$

Sign:

1. try $u=hash(\mu,i)$
2. $S^{-1}u = S^{-1}He = S^{-1}SH^+Pe = H^+Pe$
3. $\rightsquigarrow Pe$ by decode and return e as σ

Proof idea:

1. KI: $H \approx_c H'$=random
2. SD: finding e w/ H, $u=He$ is hard

[CFS01] Coutois, Finiasz, Sendrier (ASIACRYPT 2001)

Attack against CFS signature [FGO+11]

- H^+: par. mat. of a code
- $H=SH^+P$, $S \leftarrow GL(F_2, n-k)$, $P \leftarrow Perm(n)$
- $u=He$, where $e \leftarrow \chi^n$
- $vk=H$, $sk=(S,H^+,P)$

Use high-rate code to reduce #try

Sign:

1. try $u=hash(\mu,i)$
2. $S^{-1}u = S^{-1}Hx = S^{-1}SH^+Pe = H^+Pe$
3. $\rightsquigarrow Pe$ by decode and return e as σ

Proof idea:

1. KI: $H \approx_c H'$=random
2. SD: finding e w/ H,u=He is hard

[FGOPT11,13] high-rate code is distinguishable!

[FGOPT11, 13] Faugère, Gauthier-Umaña, Otmani, Perret, Tillich (ISIT2011, IEEE Trans. IT. 2013)

Summary

Code-based cryptography:

- Candidate of PQC
- Good PKE
- Bad signature

Open Problems:

- Good signature
  - Candidate: Wave [DST19]
- Good metric
  - Hamming, Rank, …
- More cryptanalysis
  - Improve ISD
  - New approach like Lattice [DDvW20]

[DST19] Debris-Alazard, Sendrier, Tillich (ASIACRYPT 2019)
[DDvW20] Debris-Alazard, Ducas, van Woerden (https://eprint.iacr.org/2020/869)