Fault-Injection Attacks against NIST's Post-Quantum Cryptography Round 3 KEM Candidates

Keita Xagawa[1], Akira Ito[2], Rei Ueno[23], Junko Takahashi[1], Naofumi Homma[24]

2021/12/06–10 ASIACRYPT 2021

1. NTT Social Informatics Laboratories, 2. Tohoku University, 3. PRESTO, JST, 4. CREST, JST

Agenda

- Why PQC?
- NIST PQC
- PKE, KEM, and FO
- Key-recovery attack using oracles
- Fault-injection analysis and skipping the equality test
- Summary

Why PQC

Post-Quantum Cryptography

NTT

Quantum Computers:

- Google: 54 qubits (2019)
- IBM Q53: 53 qubits (2019)
- USTC: 76 qubits (2020)

Countermeasure:

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

QC solves factoring/DL [Sho94]

→  may be quantum

cf. https://en.wikipedia.org/wiki/List_of_quantum_processors

[Sho94] Shor (FOCS 1994)
[BFHLV17] Brenstein, Fried, Heninger, Lou, Valenta (NIST PQC Round 1)

NIST PQC Round 3 (2020-07-22—)

4 Finalists: KEM

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

5 Alternates: KEM

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

3 Finalists: Sig

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

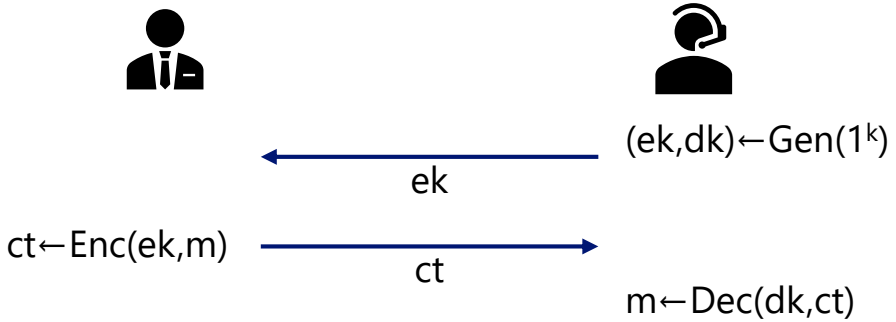
3 Alternates: Sig

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

PKE, KEM, and FO

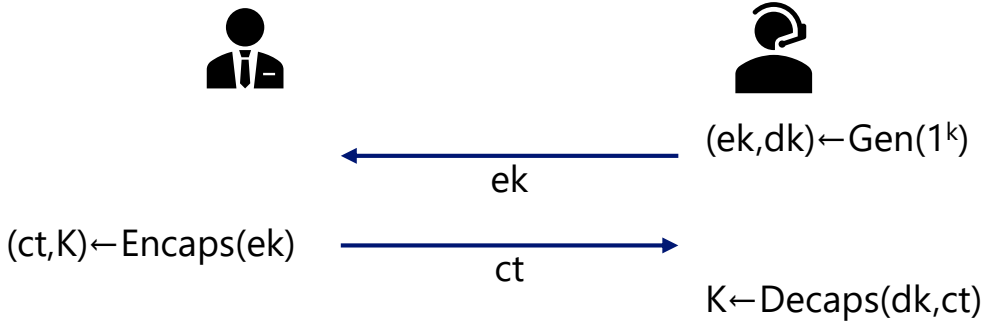
PKE – Public-Key Encryption

- $\mathrm{Gen}(1^k) \rightarrow (\mathrm{ek}, \mathrm{dk})$
- $\mathrm{Enc}(\mathrm{ek}, \mathrm{m}) \rightarrow \mathrm{ct}$
- $\mathrm{Dec}(\mathrm{dk}, \mathrm{ct}) \rightarrow \mathrm{m} / \perp$



KEM – Key Encapsulation Mechanism

- $\mathrm{Gen}(1^k) \rightarrow (\mathrm{ek}, \mathrm{dk})$
- $\mathrm{Encaps}(\mathrm{ek}) \rightarrow (\mathrm{ct}, K)$
- $\mathrm{Decaps}(\mathrm{dk}, \mathrm{ct}) \rightarrow K / \perp$



FO_{KEM} [FO99,HHK17,JZCWM18,…]

KEM = FO_{KEM}[PKE,G,H]:

- Encaps(ek;x):
  - ct=Enc(ek,x;G(x))
  - K=H(x)
- Decaps(dk,ct):
  1. x′←Dec(dk,ct)
  2. if Enc(ek,x′;G(x′))=ct
  3. then return H(x′)
  4. else return ⊥

Adapted version [FO99]:

FO_{KEM} is IND-CCA in ROM
if PKE is OW-CPA (and some)

New techniques for PQC

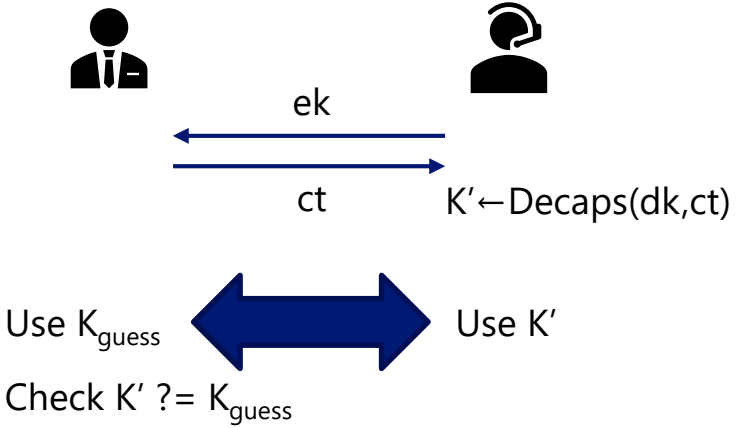
- [HHK17] Implicit Rejection
- [JZCWM18] New proof tech.
- [SXY18+HKSU20] Using DS
- [JZM19,XY19,LW21] HU, HFO
- [Zhandry19] New QRO sim.

KRA using KMO/PCO/FDO

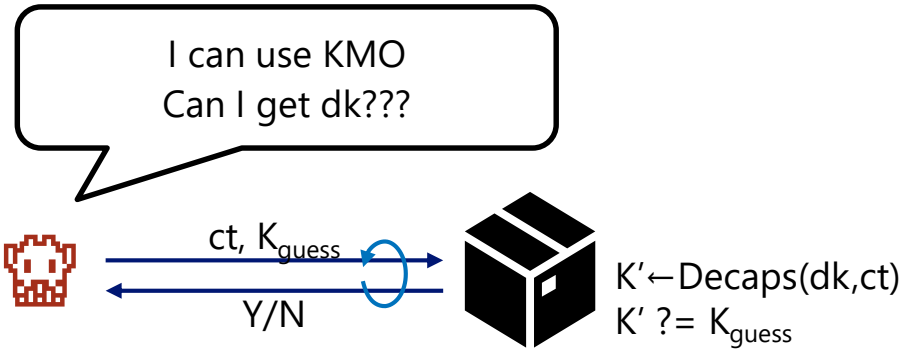
Key-Mismatch Oracle (KMO)

Key-Mismatch Oracle:

e.g., key exchange w/ fixed secret



KRA-KMO against KEM



Plaintext-Checking Oracle (PCO)

Plaintext-Checking Oracle:

KEM w/ FO and KE w/ fixed secret

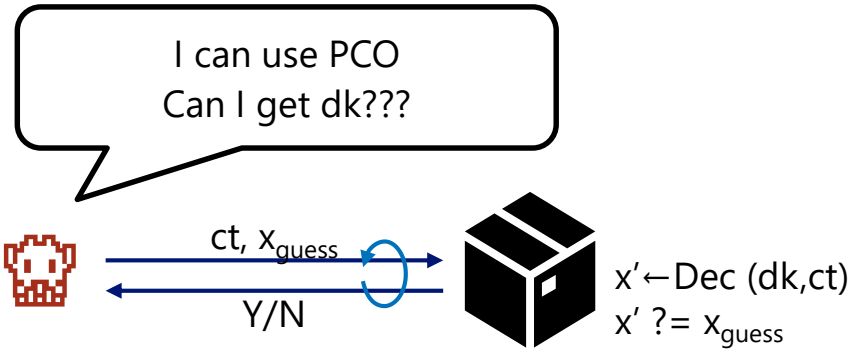
→ KMO checks $K'?=K_{guess}$

→ We can check $H(x')?=H(x_{guess})$

→ We can implement PCO of PKE

Ref: OW-PCA [OP01]

KRA-PCO against PKE



[OP01] Okamoto and Pointcheval (CTRSA 2001)

Faulty Decapsulation Oracle (FDO)

Faulty Decapsulation Oracle:

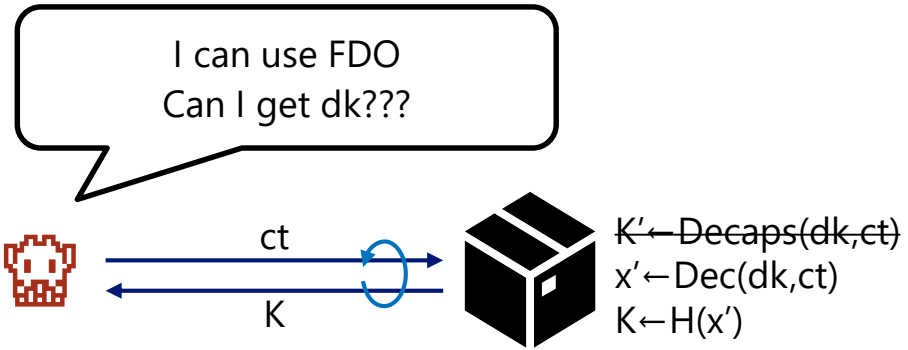
KEM w/ FO

Decapsulation *ignores* the validity test

FDO always returns H(x')

Cf. PCO ≪ FDO ≪ DO

KRA-FDO against KEM/PKE

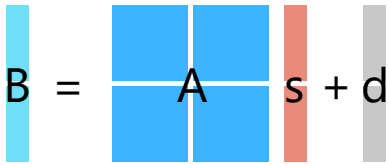


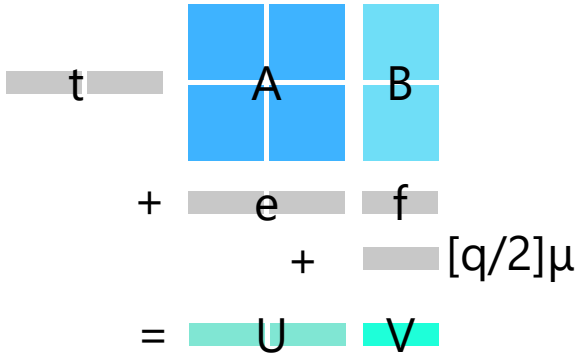
Example: Kyber

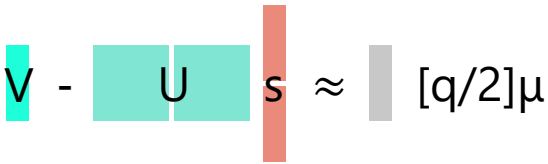
Kyber-512

Kyber-512's PKE:

- $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^{256}+1)$
- Ψ_3 = centered bin. dist. over $[-3,+3]^{256}$
- Gen(pp):
  - A←$\mathcal{R}_q^{2\times 2}$, s,d←$\Psi_3^2$, B=As+d
  - ek = (A,B), dk = s
- Enc(ek,μ; t,e,f):
  - U = tA + e, V = tB + f + [q/2]μ
  - ct=(U',V') = (comp(U),comp(V))
- Dec(dk,ct):
  1. (U,V) = (decomp(U'), decomp(V'))
  2. μ' = near((2/q) (V-Us)) mod 2



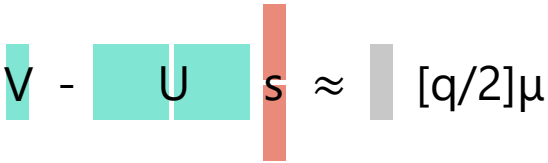




KRA-PCO against Kyber512

Kyber-512's PKE:

- $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^{256}+1)$
- Ψ_3 = centered bin. dist. over $[-3,+3]^{256}$
- Dec(dk,ct): $s \leftarrow \Psi_3^2$
  1. (U,V) = (decomp(U'), decomp(V'))
  2. μ' = near((2/q) (V-Us)) mod 2



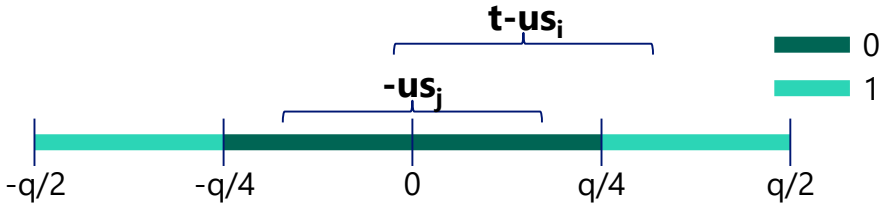
Idea [HV20]:

Consider U=(u,0), V = t x^i

→ μ_i' =near((2/q) (t - u s_i)) mod 2

and μ_j' =near((2/q) (-u s_j)) mod 2

→ Determine s_i by checking μ_i'



[HV20] Huguenin-Dumittan and Vaudenay (ACNS2020)

Behaivior of μ_i' w/ u=-276

(a) Kyber512

sk_i \ t	−3	−2	−1	0	1	2	3
−3	1	1	1	0	0	0	0
−2	1	1	0	0	0	0	0
−1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
+1	0	0	0	0	0	0	1
+2	0	0	0	0	0	1	1
+3	0	0	0	0	1	1	1

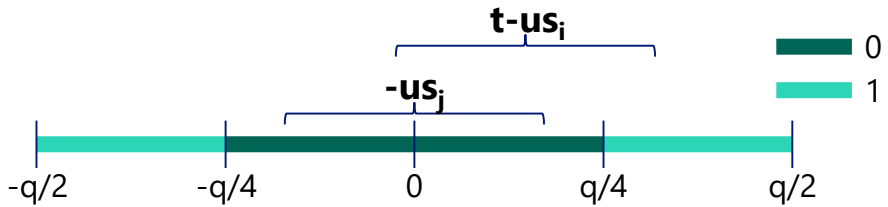
Idea [HV20]**:**

Consider U=(u,0), V = t x^i

→ μ_i' =near((2/q) (t - u s_i)) mod 2

and μ_j' =near((2/q) (-u s_j)) mod 2

→ Determine s_i by checking μ_i'



[HV20] Huguenin-Dumittan and Vaudenay (ACNS2020)

KRA-PCO – #Queries

(a) Kyber512

sk_i \ t	−3	−2	−1	0	1	2	3
−3	1	1	1	0	0	0	0
−2	1	1	0	0	0	0	0
−1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
+1	0	0	0	0	0	0	1
+2	0	0	0	0	0	1	1
+3	0	0	0	0	1	1	1

3 queries determine sk_i in [-3,+3]

sk = s←Ψ_3^2 has 256*2 coefficients

→ Need 1536 queries

[HV20] Huguenin-Dumittan and Vaudenay (ACNS2020)

FDO reduces #Queries

(a) Kyber512

sk_i \ t	−3	−2	−1	0	1	2	3
−3	1	1	1	0	0	0	0
−2	1	1	0	0	0	0	0
−1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
+1	0	0	0	0	0	0	1
+2	0	0	0	0	0	1	1
+3	0	0	0	0	1	1	1

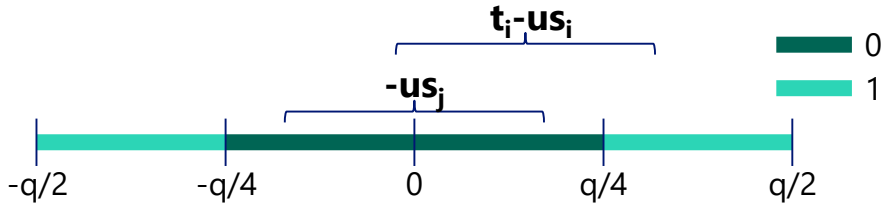
Seeing H(μ') [New]:

Consider U=(u,0), V = $t_0+t_1x+t_2x^2+t_3x^3$

→ $\mu_i' = \text{near}((2/q)\,(t_i - u\,s_i)) \bmod 2$

and $\mu_j' = \text{near}((2/q)\,(-u\,s_j)) \bmod 2$

→ Determine $s_{0,\ldots,}s_3$ by checking
H(μ') ?= H(0000 0…)/…/H(1111 0…)



KRA-FDO – #Queries

(a) Kyber512

sk_i \ t	−3	−2	−1	0	1	2	3
−3	1	1	1	0	0	0	0
−2	1	1	0	0	0	0	0
−1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
+1	0	0	0	0	0	0	1
+2	0	0	0	0	0	1	1
+3	0	0	0	0	1	1	1

3 queries determine $sk_{0,\ldots,}sk_{a-1}$

(by checking 2^a candidates)

sk = s←Ψ_3^2 has 256*2 coefficients

→ Need 1536/a queries

[HV20] Huguenin-Dumittan and Vaudenay (ACNS2020)

Survey of KRA-PCO

KEM	Ref.
Kyber	[QCD19,RRCB20,HV20,RBRC20,QCZ+21]
Saber	[HV20,OUKT21,NDGJ21,QCZ+21]
FrodoKEM	[BDH+19,RRCB20,VV20,QCZ+21]
NTRU LPRime	[New:XIU+21]
NTRU-HPS	[DDS+19] (There are older ones [JJ00,HS00,…])
NTRU-HRSS	[ZCQD21]
Streamlined NTRU Prime	[REB+21]

See [App.C:XIU+21] for attacks against other schemes

Fault-Injection Analysis

Fault-Injection Analysis

FO_{KEM}

- Decaps(dk,ct):
  1. $x' \leftarrow Dec(dk,ct)$
  2. if $Enc(ek,x';G(x'))=ct$
  3. then return $H(x')$
  4. else return $\perp$

Skip the equality check

- Decaps_fail(dk,ct):
  1. $x' \leftarrow Dec(dk,ct)$
  2. ~~if $Enc(ek,x';G(x'))=ct$~~
  3. then return $H(x')$
  4. ~~else return $\perp$~~

Skip step 2 by injecting power glitch
→We have FDO virtually!

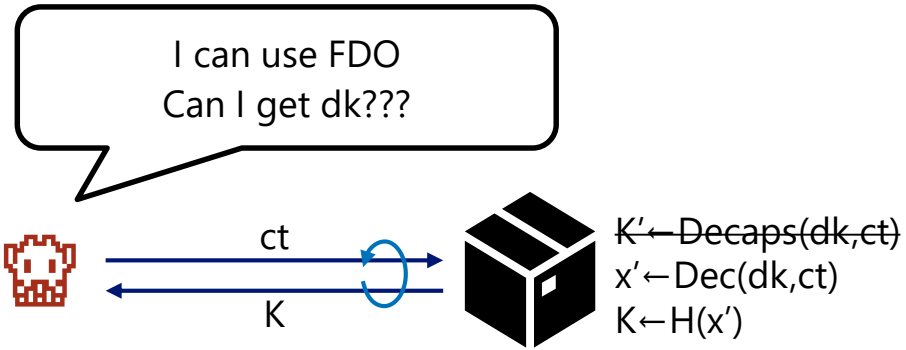
Fault-Injection Analysis

Skip the equality check

- Decaps_fail(dk,ct):
  1. x′←Dec(dk,ct)
  2. ~~if Enc(ek,x′;G(x′))=ct~~
  3. then return H(x′)
  4. ~~else return H′(s,ct) //PRF~~

Skip step 2 by injecting power glitch
→We have FDO virtually!

KRA with FDO:



CCA Bug in NTRU Prime (A.Ito found)

```
1087  /* 0 if matching ciphertext+confirm, else -1 */
1088  /**************************************************
1089  * Name:        Ciphertexts_diff_mask
1090  *
1091  * Description: Returns 0 if ciphertexts and Confirm bytes, -1 otherwise
1092  *
1093  * Arguments:
1094  * const unsigned char *c  : pointer to the input first ciphertext
1095  * const unsigned char *c2 : pointer to the input second ciphertext
1096  **************************************************/
1097  static int Ciphertexts_diff_mask(const unsigned char *c,const unsigned char *c2)
1098  {
1099    uint16 differentbits = 0;
1100    int len = Ciphertexts_bytes+Confirm_bytes;
1101
1102    int *cc = (int *)(void *)c;
1103    int *cc2 = (int *)(void *)c2;
1104    int differentbits2 = 0;
1105    for (len-=4 ;len>=0; len-=4) {
1106      differentbits2 = __USADA8((*cc++),(*cc2++),differentbits2);
1107    }
1108    c = (unsigned char *)(void *) cc;
1109    c2 = (unsigned char *)(void *) cc2;
1110    for (len &= 3; len > 0; len--)
1111      differentbits2 =__USADA8((*c++),(*c2++),differentbits2);
1112    return ((-1)-((differentbits-1)>>31));
1113  }
```

Compare c and c2

Return 0 if c = c2, -1 otherwise

https://github.com/mupq/pqm4/blob/master/crypto_kem/sntrup761/m4f/kem.c

CCA Bug in NTRU Prime (A.Ito found)

```
1087  /* 0 if matching ciphertext+confirm, else -1 */
1088  /************************************************
1089  * Name:        Ciphertexts_diff_mask
1090  *
1091  * Description: Returns 0 if ciphertexts and Confirm bytes, -1 otherwise
1092  *
1093  * Arguments:
1094  * const unsigned char *c  : pointer to the input first ciphertext
1095  * const unsigned char *c2 : pointer to the input second ciphertext
1096  ************************************************/
1097  static int Ciphertexts_diff_mask(const unsigned char *c,const unsigned char *c2)
1098  {
1099    uint16 differentbits = 0;
1100    int len = Ciphertexts_bytes+Confirm_bytes;
1101
1102    int *cc = (int *)(void *)c;
1103    int *cc2 = (int *)(void *)c2;
1104    int differentbits2 = 0;
1105    for (len-=4 ;len>=0; len-=4) {
1106      differentbits2 = __USADA8((*cc++),(*cc2++),differentbits2);
1107    }
1108    c = (unsigned char *)(void *) cc;
1109    c2 = (unsigned char *)(void *) cc2;
1110    for (len &= 3; len > 0; len--)
1111      differentbits2 =__USADA8((*c++),(*c2++),differentbits2);
1112    return ((-1)-((differentbits-1)>>31));
1113  }
```

return 0 if c = c2 else -1

L1112: return (-1) – ((d-1)>>31):

this returns 0 if d=0 else -1

- d-1 = 0xffff_ffff/non-zero
- (d-1)>>31 = -1/0
- (-1) – ((d-1)>>31) = 0/-1

L1099: uint16 d = 0x0000

differentbits remains 0
while differentbits2 is changed
→this function always returns 0

https://github.com/mupq/pqm4/blob/master/crypto_kem/sntrup761/m4f/kem.c

Fixed!

A bug in decasuplation function of NTRU Prime implementation #195

Closed **arokiti** opened this issue on 21 Jun · 1 comment

arokiti commented on 21 Jun

Hi. I found a bug in the decapsulation function of the NTRU Prime implementation. Specifically, the "Ciphertexts_diff_mask" function in kem.c of sntrup761 returns a mask value as follows:

```
1112:   return ((-1)-((differentbits-1)>>31));
```

However, this function always returns 0 because the variable "differentbits" is always 0. Probably, "differentbits2" is correct, not "differentbits". Therefore, I think it can be fixed by replacing

```
1112:   return ((-1)-((differentbits-1)>>31));
```

with

```
1112:   return ((-1)-((differentbits2-1)>>31));
```

mkannwischer added a commit that referenced this issue on 23 Jul

`Fixes #195: Bug in ntrup761/ntrulpr761 decaps.` `847ae68`

mkannwischer mentioned this issue on 23 Jul

Fixes bug in ntrup761/ntrulpr761 decapssulation #200 Merged

mkannwischer commented on 23 Jul Contributor

Sorry for the super long delay.
Thanks for reporting this obvious bug. I agree with your fix and it will shortly be merged.

mkannwischer closed this in `b4c013e` on 23 Jul

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked pull requ
Successfully mer
None yet

Notifications
You're not receiv

2 participants

https://github.com/mupq/pqm4/issues/195

Timing Bug in FrodoKEM [GJN20]

```
228       // Is (Bp == BBp & C == CC) = true
229       if (memcmp(Bp, BBp, 2 * PARAMS_N * PARAMS_NBAR) == 0 && memcmp(C, CC, 2 * PARAMS_NBAR * PARAMS_NBAR) == 0) {
230           // Load k' to do ss = F(ct || k')
231           memcpy(Fin_k, kprime, CRYPTO_BYTES);
232       } else {
233           // Load s to do ss = F(ct || s)
234           memcpy(Fin_k, sk_s, CRYPTO_BYTES);
235       }
236       shake(ss, CRYPTO_BYTES, Fin, CRYPTO_CIPHERTEXTBYTES + CRYPTO_BYTES);
```

Idea of [GJN20]:
If Bp != BBp, it returns ss = F(ct,s) **rapidly**

Let ct = (Bp,C) = Enc(pk,μ; G(μ))
We can check if ct' =(Bp,C') is decrypted into μ or not by seeing the timing
→ lead to key-recovery attack

[GJN20] Guo, Johansson, Nilsson (CRYPTO 2020)

Fixed!

Fix timing leakage in FrodoKEM (and others?) #161

Closed · Ko- opened this issue on 27 Jun 2020 · 1 comment

Ko- commented on 27 Jun 2020 — Contributor

In this CRYPTO paper a timing attack on some implementations of the FO transform was identified. This affects decapsulation in FrodoKEM, which was fixed in microsoft/PQCrypto-LWEKE@ `155c24c` and PQClean/PQClean@ `ae1530d`. It doesn't change the test vectors or anything, but it's probably best to update pqm4 as well. The `m4` implementations are also affected. Although I expect the impact on benchmarks to be negligible, we should probably rerun the benchmarks.

It's not clear to me at this point which other implementations that do claim constant-time-ness require changes.

kriskwiatkowski commented on 15 Aug 2020 — Contributor

if you still consider sike then it has similar issue (but not exactly the same)

pqm4/crypto_kem/sikep434/m4/sike.inc
Line 87 in 56417d9

```
87    if (memcmp(c0_, ct, CRYPTO_PUBLICKEYBYTES) != 0) {
```

mkannwischer added a commit that referenced this issue on 15 Sep

Fix #161. … 4836e45

mkannwischer mentioned this issue on 15 Sep

Eliminate memcmp in FrodoKEM #214 — Merged

rpls closed this in #214 on 27 Sep

rpls added a commit that referenced this issue on 27 Sep

Fix #161. … 9c7be01

Assignees: No one assigned

Labels: None yet

Projects: None yet

Milestone: No milestone

Linked pull requ… Successfully mer… Eliminate me…

Notifications: You're not receiv…

2 participants

Eliminate memcmp in FrodoKEM #214

Merged · rpls merged 1 commit into `master` from `frodofix` on 27 Sep

Conversation 1 · Commits 1 · Checks 0 · Files changed 3

Changes from all commits · File filter · Conversations · Jump to

Fix #161.

```
I wish people would just submit a patch to pqm4 instead of writing
another paper about a bug that is well known...
Anyway, I fixed this now...
```

mkannwischer committed on 15 Sep

https://github.com/mupq/pqm4/issues/161

Skip of cmov – Saber, Kyber, NTRU

```
1     int crypto_kem_dec(uint8_t *k, const uint8_t *c, const uint8_t *sk)
2 {
3     uint8_t fail;
4     uint8_t buf[64];
5     uint8_t kr[64]; // Will contain key, coins
6     const uint8_t *pk = sk + SABER_INDCPA_SECRETKEYBYTES;
7     const uint8_t *hpk = sk + SABER_SECRETKEYBYTES - 64;
8                                     // Save hash by storing h(pk) in sk
9
10    indcpa_kem_dec(sk, c, buf);
11    memcpy(buf + 32, hpk, 32);
12    sha3_512(kr, buf, 64);          // kr = G(x',H(pk))
13    fail = indcpa_kem_enc_cmp(buf, kr + 32, pk, c);
14    sha3_256(kr + 32, c, SABER_BYTES_CCA_DEC);    // kr = G1(x',H1(pk)) + H1(ct)
15    cmov(kr, sk + SABER_SECRETKEYBYTES - SABER_KEYBYTES,
16                                                  SABER_KEYBYTES, fail);
17    sha3_256(k, kr, 64);
18    return (0);
19 }
20
```

L15: cmov: the half of kr is $G_1(x',H_1(pk))$ if fail is 0, seed otherwise
L17: sha3_256: k = $H(G_1(x',H_1(pk)),H_1(ct))$ if fail is 0, $H(seed,H_1(ct))$ otherwise

https://github.com/mupq/pqm4/blob/master/crypto_kem/saber/m4fspeed/kem.c

Skip of cmov – Saber, Kyber, NTRU

```
1   int crypto_kem_dec(uint8_t *k, const uint8_t *c, const uint8_t *sk)
2   {
3       uint8_t fail;
4       uint8_t buf[64];
5       uint8_t kr[64]; // Will contain key, coins
6       const uint8_t *pk = sk + SABER_INDCPA_SECRETKEYBYTES;
7       const uint8_t *hpk = sk + SABER_SECRETKEYBYTES - 64;
8                                   // Save hash by storing h(pk) in sk
9
10      indcpa_kem_dec(sk, c, buf);
11      memcpy(buf + 32, hpk, 32);
12      sha3_512(kr, buf, 64);              // kr = G(x',H(pk))
13      fail = indcpa_kem_enc_cmp(buf, kr + 32, pk, c);
14      sha3_256(kr + 32, c, SABER_BYTES_CCA_DEC);   // kr = G1(x',H1(pk)) + H1(ct)
15      cmov(kr, sk + SABER_SECRETKEYBYTES - SABER_KEYBYTES,
16                                                  SABER_KEYBYTES, fail);
17      sha3_256(k, kr, 64);
18      return (0);
19  }
20
```

```
1       bl   sha3_256
2   .LVL26:
3   .loc 1 79 3 is_stmt 1 view .LVU62
4       uxtb    r3, r7
5       add r1, r4, #1536
6       add r0, sp, #64
7       movs    r2, #32
8       bl   cmov
9   .LVL27:
10      .loc 1 82 3 view .LVU63
11      movs    r2, #64
12      mov r0, r6
13      add r1, sp, r2
14      bl   sha3_256
```

L15: cmov: the half of kr is $G_1(x',H_1(pk))$ if fail is 0, seed otherwise
L17: sha3_256: k = $H(G_1(x',H_1(pk)),H_1(ct))$ if fail is 0, $H(seed,H_1(ct))$ otherwise

https://github.com/mupq/pqm4/blob/master/crypto_kem/saber/m4fspeed/kem.c

Skip of cmov – Saber, Kyber, NTRU

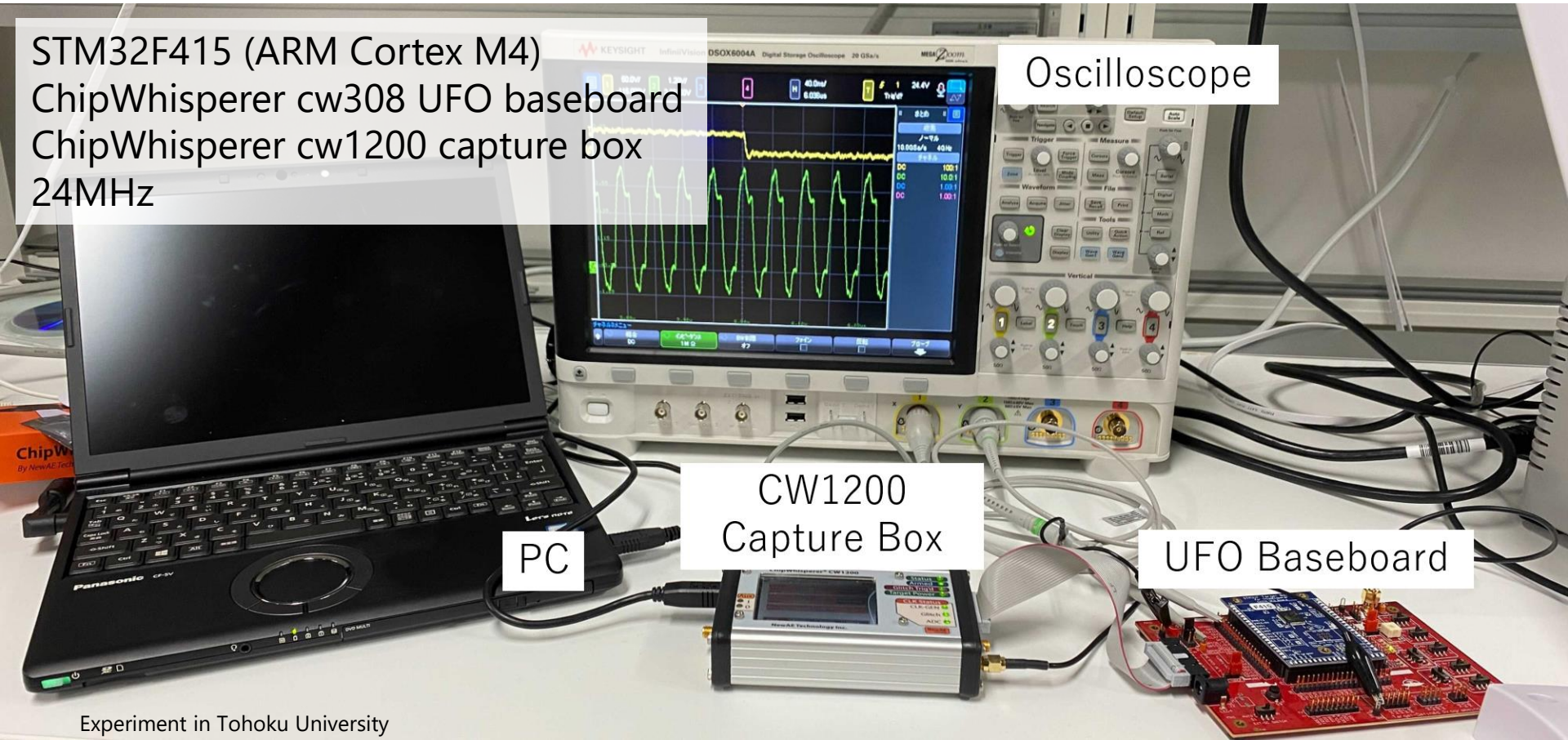
```
1     int crypto_kem_dec(uint8_t *k, const uint8_t *c, const uint8_t *sk)
2 {
3     uint8_t fail;
4     uint8_t buf[64];
5     uint8_t kr[64]; // Will contain key, coins
6     const uint8_t *pk = sk + SABER_INDCPA_SECRETKEYBYTES;
7     const uint8_t *hpk = sk + SABER_SECRETKEYBYTES - 64;
8                                   // Save hash by storing h(pk) in sk
9
10    indcpa_kem_dec(sk, c, buf);
11    memcpy(buf + 32, hpk, 32);
12    sha3_512(kr, buf, 64);              // kr = G(x',H(pk))
13    fail = indcpa_kem_enc_cmp(buf, kr + 32, pk, c);
14    sha3_256(kr + 32, c, SABER_BYTES_CCA_DEC);   // kr = G1(x',H1(pk)) + H1(ct)
15    cmov(kr, sk + SABER_SECRETKEYBYTES - SABER_KEYBYTES,
16                                                SABER_KEYBYTES, fail);
17    sha3_256(k, kr, 64);
18    return (0);
19 }
20
```

```
1     bl   sha3_256
2 .LVL26:
3 .loc 1 79 3 is_stmt 1 view .LVU62
4     uxtb    r3, r7
5     add r1, r4, #1536
6     add r0, sp, #64
7     movs    r2, #32
8     bl   cmov
9 .LVL27:
10    .loc 1 82 3 view .LVU63
11    movs    r2, #64
12    mov r0, r6
13    add r1, sp, r2
14    bl   sha3_256
```

L15: cmov: the half of kr is $G_1(x',H_1(pk))$ if fail is 0, seed otherwise
L17: sha3_256: k = $H(G_1(x',H_1(pk)),H_1(ct))$ if fail is 0, $H(seed,H_1(ct))$ otherwise
By skipping L8 in RHS, k = $H(G_1(x',H_1(pk)),H_1(ct))$ *always*

Experiment

STM32F415 (ARM Cortex M4)
ChipWhisperer cw308 UFO baseboard
ChipWhisperer cw1200 capture box
24MHz



Experiment in Tohoku University

Experimental results

100 skip trials / each trial takes 0.1s – 10s

KEM	# failures	# successes	# exp. queries
Kyber512	60	52	5908※
LightSaber	74	46	15515※
ntruhps2048509	33	33	2235
Bikel1	49	34	-
sikep434	30	15	1787
cf. ntrulpr653	100	100	1306※

※: We can reduce #queries for Kyber, Saber, and ntrulpr

Summary

KEM	PCO/FDO	Attack surface in pqm4	Effect of FIA in pqm4
Classic McEliece	?	N/A	N/A
Kyber	KR	Skip	KR
NTRU	KR	Skip	KR
Saber	KR	Skip	KR
BIKE	KL (new)	Skip	KL
FrodoKEM	KR	Timing bug	KR
HQC	KR	N/A	N/A
NTRU Prime (sntrupr)	KR	CCA bug	KR
NTRU Prime (ntrulpr)	KR (new)	CCA bug	KR
SIKE	KR	Skip	KR

KR=Key Recovery, KL=Key leakage

Summary

- Survey of KRA-PCO/FDO
  - FDO setting reduces # queries for KYBER, Saber, FrodoKEM, NTRU LPRime
- Fault-injection analysis and skipping the equality test
- Open problems
  - KRA-PCO against Classic McEliece
  - KRA-PCO against BIKE
  - Secure implementations