Post-Quantum Backdoor for Kyber-KEM

Reporter: Haoxiang Jin

Authors: Wenwen Xia[1,2], Geng Wang[3,2,*], Dawu Gu[3,2,1,*]

1 School of Cyber Engineering, Xidian University, Xi'an, 710071, China xiawenwen@stu.xidian.edu.cn

2 Lab of Cryptology and Computer Security, Shanghai Jiao Tong University, Shanghai, 200240, China

3 School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China {wanggxx, dwgu}@sjtu.edu.cn

* Corresponding author

Background

3. Development of Kleptography

[YY97]
Firstly proposed kleptography.

[KLT17]
Backdoor Embedding to NTRU encryptsystem
- The first backdoor for post-quantum cryptographic algorithm.

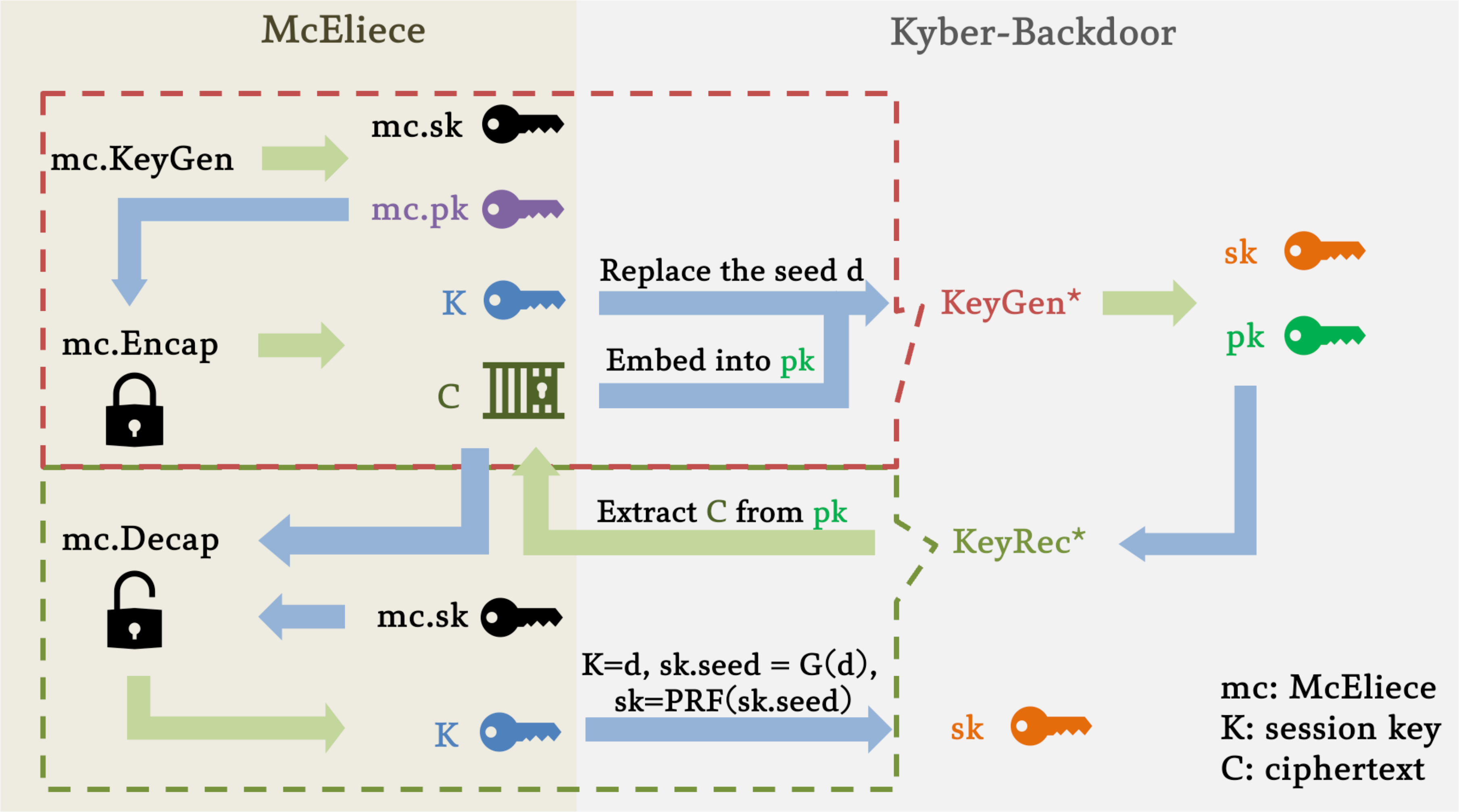
[XY18,YCL+20]
Backdoor for LWE-based cryptsystem
- General backdoor construction for LWE-based cryptsystem.
- Drawback: Cannot Apply to IND-CCA2 post-quantum KEM.

[YXP20]
Backdoor for New Hope KEM
- General backdoor construction for LWE-based cryptsystem.
- Drawback: Use elliptic curve-based Diffie-Hellman key exchange as a backdoor, lack of post-quantum undetectability.

[Hem20]
Backdoor for New Hope KEM
- Fix the construction flaw in [YXP20].
- Drawback: Lack of post-quantum undetectability.

[RBC+24]
Post-quantum backdoor for Kyber
- Claim to be publicly undetectable, but is not satisfied.
- Drawback: Can be detected by Kyber private key holders

Roadmap



Post-Quantum Backdoor for Kyber-KEM

Basic Knowledge

Public Undetectibility

Challenger $\mathcal{C}$ | Detector $\mathcal{D}$

Randomly choose $b \leftarrow \{0,1\}$.
If $b = 0$, run (sk,pk)=KeyGen*.
If $b = 1$, run (sk,pk)=KeyGen.

pk →

← M

Choose message $\mathrm{M} \leftarrow \{0,1\}^l$.

If $b = 0$, run C=Enc*(M,sk).
If $b = 1$, run C=Enc(M,sk).

C →

Output b'.

Pr(b = b') − 1/2 is negligible.

Strict Undetectibility

Challenger $\mathcal{C}$

Detector $\mathcal{D}$

Randomly choose $b \leftarrow \{0,1\}$.
If $b = 0$, run (sk,pk)=KeyGen*.
If $b = 1$, run (sk,pk)=KeyGen.

pk, sk →

Choose message $\mathrm{M} \leftarrow \{0,1\}^{l}$
and ask $\mathcal{C}$ run Enc,
or ask $\mathcal{C}$ run Encap.

← (M, Enc) or Encap

If Enc and $b = 0$, C=Enc*(M,sk).
If Enc and $b = 1$, C=Enc(M,sk).
If Encap, (K, C)=Encap(pk).

C →

Output b'.

Pr(b = b') − 1/2 is negligible.

McEliece KEM

- **Key Generation (mc.KeyGen)**

  Generate a key pair $(mc.pk, mc.sk)$, where ublic key is a matrix $\mathbf{T} \in \{0,1\}^{(m_1 \cdot t) \times k}$.

- **Encapsulation (mc.Encap)**

  1. Input $mc.pk = \mathbf{T}$, generate a binary vector $\mathbf{v} \in \{0,1\}^n$ of weight $wt(\mathbf{v}) = t$.
  2. Compute ciphertext $C = \mathrm{ENCODE}(\mathbf{v}, mc.pk) = (\mathbf{I}|\mathbf{T}) \cdot \mathbf{v}$.
  3. Compute the session key $K = H(1, \mathbf{v}, C)$.
  4. Output (C, K).

- **Decapsulation (mc.Decap)**

  1. Compute $\mathbf{v} = \mathrm{DECODE}(C, mc.sk)$.
  2. Compute and output $K = H(1, \mathbf{v}, C)$.

- In McEliece348864, $m_1 = 12, t = 64, k = 2720, n = m_1 \cdot t + k = 3488$, thus the ciphertext size $m_1 t = 768$.

Construct Backdoor of Kyber through McEliece (KeyGen*)

```
output: pk ← (t, pk.seed), sk ← s
Function Kyber.KeyGen():
    d ← B^32.
    (sk.seed, pk.seed) ← G(d) //Hash Function G is declared in Kyber
    (s, e) ← PRF(sk.seed) //Sample s and e from sk.seed in distribution B_η
    A ← Parse(XOF(pk.seed)) //Sample A from pk.seed .
    t ← As + e mod± q;
    return pk ← (t, pk.seed), sk ← s
```

Algorithm 1: Kyber Key Generation Algorithm KeyGen

- Replace d with session key K generated from McEliece

B_η -- Central Binomial Distribution:

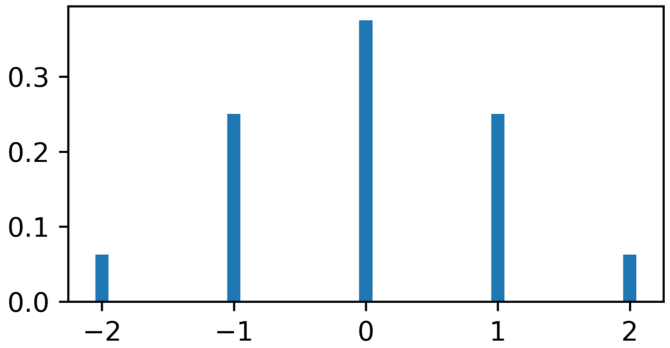
Sample

$$(a_1, \dots, a_\eta, b_1, \dots, b_\eta) \leftarrow \{0,1\}^{2\eta}$$

and output $\sum_{i=1}^{\eta}(a_i - b_i)$

Kyber512: $\eta = 3$

Kyber768 and Kyber 1024: $\eta = 2$



- Embed $C = \mathrm{ENCODE}(\boldsymbol{v}, \mathrm{mc.pk})$ from McEliece into LSB($\mathbf{t}$) by sampling a special **e** following the same distribution while ignoring border case of t_i.
- Suppose the backdoor user has mc.sk, then he can decrypt the seed d after receiving pk = (t, pk.seed) by computing d' = mc.Decap(mc.sk, LSBs(t)).
- Here $\boldsymbol{v} = \mathrm{DECODE}(C, mc.sk)$, $K = H(1, \boldsymbol{v}, C)$

How to do this?

Construct Backdoor of Kyber through McEliece (KeyGen*)

- Sample a special **e** following the same distribution:

Kyber768 and Kyber 1024: $\eta = 2$, then e_i follows distribution B_2 as:

Value	-2	-1	0	1	2
Probability	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{1}{4}$	$\frac{1}{16}$

$\Pr(\mathrm{LSB}(e_i) = 0) = \Pr(\mathrm{LSB}(e_i) = 1) = \frac{1}{2}$.

Depart the probabilistic distribution of B_2 into two distributions:

D_1 with $\mathrm{LSB}(e_i) = 0$

Value	-2	0	2
Probability	$\frac{1}{8}$	$\frac{3}{4}$	$\frac{1}{8}$

D_1 with $\mathrm{LSB}(e_i) = 1$

Value	-1	1
Probability	$\frac{1}{2}$	$\frac{1}{2}$

Use reject sampling based on centered binomial distribution B_2

Construct Backdoor of Kyber through McEliece (KeyGen*)

input : mc.pk
output: $pk \leftarrow (\mathbf{t}, pk.seed)$, $sk \leftarrow \mathbf{s}$

Function KeyGen*(mc.pk):
    $(K, C) \leftarrow$ mc.Encap(mc.pk)
    $d \leftarrow K$ // Let the seed in Kyber be the session key of McEliece.
    $(sk.seed, pk.seed) \leftarrow G(d)$ //Function G is declared in Kyber
    $(\mathbf{s}, \_) \leftarrow \mathsf{PRF}(sk.seed)$ //Sample $\mathbf{s}$ from $sk.seed$ in distribution B_η
    $\mathbf{A} \leftarrow \mathsf{Parse}(\mathsf{XOF}(pk.seed))$ //Sample $\mathbf{A}$ from $pk.seed$.
    $\mathbf{t} \leftarrow \mathbf{As}$;
    for i **from** 1 **to** $\dim(\mathbf{t})$ **do**
        if $i \leq \mathrm{len}(C)$ **then**
            if $(\mathbf{t}[i] - C[i]) \mod 2 = 1$ **then**
                Sample e_i from the probabilistic distribution $\mathcal{D}_1$
            else
                Sample e_i from the probabilistic distribution $\mathcal{D}_0$
        else
            Sample e_i from the probabilistic distribution B_2
        $\mathbf{t}[i] \leftarrow \mathbf{t}[i] + e_i \mod^{\pm} q$
    return $pk \leftarrow (\mathbf{t}, pk.seed)$, $sk \leftarrow \mathbf{s}$

Algorithm 2: Backdoor Key Generation Algorithm KeyGen*

Replace seed d with session K

Embed C into LSB($\boldsymbol{t}$)

Strict Undetectability of our Backdoor

Lemma 1. *If C is uniformly distributed and independent with $\mathbf{A}$, $\mathbf{s}$, then the distribution of $\mathbf{e}$ generated from Algorithm 2 is also independent with $\mathbf{A}$, $\mathbf{s}$, and identical with random $\mathbf{e}$ where each coefficient is randomly sampled from B_2.*

Theorem 1. *The backdoor scheme is strictly undetectable.*

Backdoor Key Recovery (KeyRec*)

- **Discussion on the border case.**
  - $\mathrm{LSB}(t_i)$ follows uniform distribution on $\mathbb{Z}_q$ for $q = 3329$ actually. Thus,

$$\Pr(LSB(t_i) = 0) = \frac{1665}{3329} = \frac{1}{2} + \frac{1}{6658}.$$

  In border case, the recovery of C_i might fail. For example,

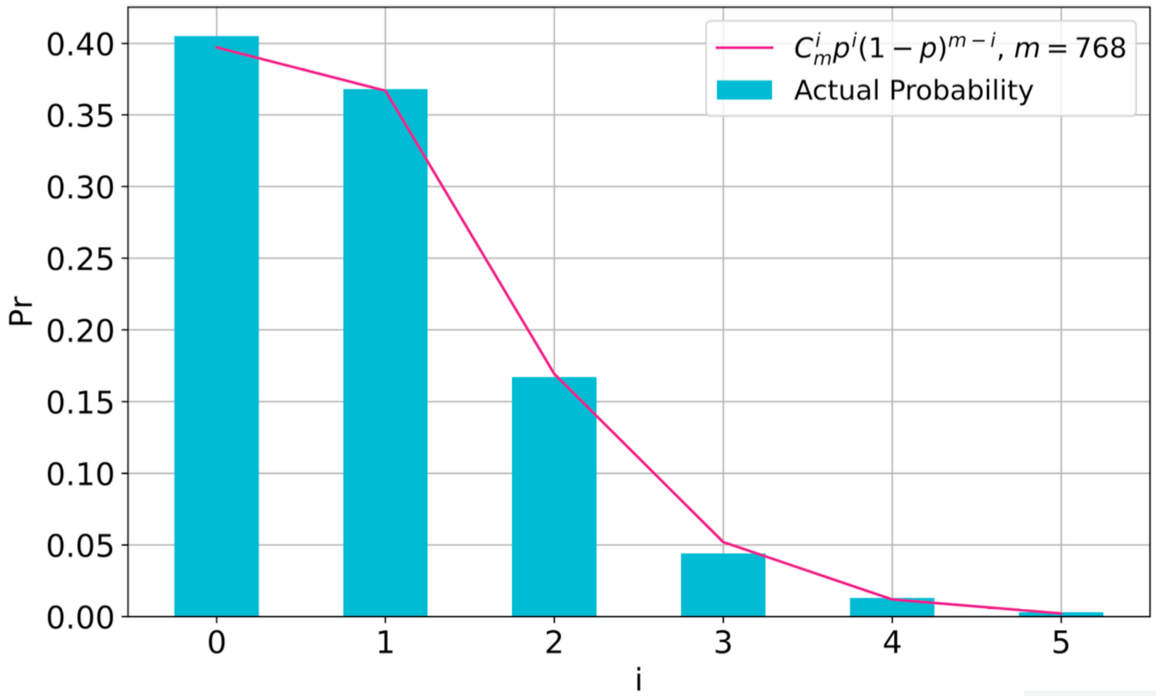
$$\left(\frac{q-1}{2}(\mathrm{mod}^{\pm}\, q)\right)(\mathrm{mod}\ 2) = \left(\frac{q-1}{2} + 1(\mathrm{mod}^{\pm}\, q)\right)(\mathrm{mod}\ 2) = 0.$$

  - $\mathrm{LSB}(t_i)$ and C_i disagree only when $t_i \in \{-\frac{q-1}{2}, -\frac{q-3}{2}, \frac{q-3}{2}, \frac{q-1}{2}\}$, so $p = \Pr(\mathrm{LSB}(t_i)$ and C_i disagree$) = \frac{4}{q}$.
  - For $q = 3329$ in Kyber, the probability that i border case elements occurence is

$$P_{\mathrm{theo}}(i) = \Pr\big(i \text{ border case elements in } (t_1, \ldots, t_m)\big) = C_m^i p^i (1-p)^{m-i},$$

  where $m = 768$ is the bit size of McEliece348864 ciphertext C.
  - The probability that there are more than 4 border case elements is only about $P_{\mathrm{theo}}(i) \approx$0.2%.

Backdoor Key Recovery (KeyRec*)



Border case probability among $m = 768$ elements

- x-axis is the number of border case elements among m elements.
- Test 1000 Kyber768 instances.
- The result of Kyber1024 is close to Kyber768 since the bit size of McEliece ciphertext is same.
- The accuracy of P_{theo} fits well to P_{actual}.
- The border case probability decreases rapidly with the growth of border case number i.

$$P_{\text{theo}}(i) = \Pr\big(i \text{ border case elements in } (t_1, \dots, t_m)\big) = C^i_m p^i (1-p)^{m-i}$$

$$P_{\text{actual}}(i) = \frac{i \text{ border case elements occur in } (t_1, \dots, t_m)}{1000}$$

Backdoor Key Recovery (KeyRec*)

input : $pk \leftarrow (\mathbf{t}, pk.seed)$, mc.$sk$, $\eta \leftarrow 2$
output: $sk \leftarrow \mathbf{s}$

Function KeyRec*(pk):
  Sample $\mathbf{A}$ from $pk.seed$
  $C' \leftarrow \text{LSBs}(\mathbf{t})$, mark $C'[i] = \star$ if $\mathbf{t}[i] \geq (q-3)/2$ or $\mathbf{t}[i] \leq -(q-3)/2$
  repeat
    $d' \leftarrow \text{mc.Decap}(\text{mc}.sk, C')$
    $(sk.seed', pk.seed') \leftarrow G(d')$
    if $pk.seed' = pk.seed$ **then**
      $(\mathbf{s}', \_) \leftarrow \mathsf{PRF}(sk.seed')$ //Sample $\mathbf{s}'$ from $sk.seed'$ through pseudorandom function PRF
      return $sk \leftarrow \mathbf{s}'$
  until *Set $C'[i] = \star$ to 0 or 1 respectively and exhaust all possibilities*;
  return $\perp$

Algorithm 3: Backdoor Key Recovery Algorithm KeyRec*

Enumerate border case.

Efficiency Test of KeyGen* and KeyRec*

方案	Cost Type (cycles/tick)	KeyGen	KeyGen*	KeyRec*
Kyber768	Median Cost/s	28397	115590	166088
	Average Cost/s	36207	118271	169267
Kyber1024	Median Cost/s	39636	133840	191503
	Average Cost/s	48604	135736	194552

- We have implemented our backdoor embedding method in C language in open source code: https://github.com/Summwer/kyber-backdoor
- All experiments were ran on a single core (Intel(R) Core(TM) i5-9500 CPU @ 3.00GHz).
- Each experimental result is median/averaged over 1000 instances.
- We achieve a 100% success rate in Kyber secret key recovery.

Possible Fixes for Backdoor

(Resistant to strict undetectability) A possible fix for [YXP20] type backdoor.

- Add seed d into the secret key.
- Secret key holder can firstly generate $pk.seed$ and $sk.seed$ from d, then compute

$$\boldsymbol{A} = \text{Parse}\big(\text{XOF}(pk.seed)\big),$$
$$(\mathbf{s}, \mathbf{e}) = \text{PRF}(sk.seed).$$

- The secret key holder determines whether the algorithm has been added to the backdoor by verifying whether the following equation holds:

$$\mathbf{As} + \mathbf{e} = \mathbf{t} \bmod^{\pm} q.$$

If the equation doesn't hold, then there is a backdoor in the scheme.

- This method can be used to fix the backdoor construction scheme proposed by [YXP20, Hem22] and our backdoor scheme.

- Even with the fix method on the left, the backdoor of this article and [ZXP20, Hem22] is still publicly undetectable.
- [ZXP20, Hem22] is a backdoor construction scheme based on elliptic curves.

(Resistant to public undetectability) A possible fix for [YXP20, Hem22].

- crs: the common reference string generated by a trusted method (e.g. MPC protocol).
- Each user's public key seed is generated by $pk.seed = H(\text{crs} \parallel id)$, in which id is the identity of a user, $H(\cdot)$ is a hash function.
- Since the generation method of $pk.seed$ is known, it is easy for users to find out if it is replaced.
- Since our backdoor doesn't modify pk.seed, it is not affected.

Comparison with previous backdoors on post-quantum schemes

Work	Post-Quantum	Valid for KEM	Undetectability	Provable
Kwant et al[KLT17]	X	X	X	N/A
Xiao and Yu [XY18]	√	X	√	X
Yang et al [YCL+20]	√	X	√	√
Yang et al [YXP20]	X	√	X	N/A
Hemmert [Hem22]	X	√	√	√
Ravi et al [RBC+22]	√	√	X	N/A
This Work	√	√	√	√

- "Post-Quantum": Backdoor construction is based on a Post-Quantum public key cryptsystem.
- "Undetectability": Undectectability of each work.
- "Provable": A formal proof of undetectability is provided.

Thanks!

If you have any questions, please contact to email xiawenwen@stu.xidian.edu.cn