



# Post-Quantum Cryptography Standards

Spiegazione e Implementazione del KEM CRYSTALS-Kyber

Sicurezza dei Sistemi e delle Reti Informatiche

**Samuele Manclossi** (09882A)

July 14, 2024



UNIVERSITÀ  
DEGLI STUDI  
DI MILANO



*I recenti miglioramenti nella tecnologia del **Quantum Computing** hanno accelerato la ricerca di soluzioni alternative alla fattorizzazione come **problema difficile**, ricerca già iniziata con la pubblicazione dell'**Algoritmo di Shor**.*

*Questa ricerca si è tramutata tra Agosto e Dicembre 2016 in una gara del **NIST** per ottenere dei nuovi standard. Nel 2017 è stato proposto il sistema che vedremo e nel **2021** esso è diventato formalmente un **nuovo standard**.*



# Table of Contents

## 1 Key Exchange and Quantum Security

- ▶ Key Exchange and Quantum Security
- ▶ CRYSTALS-Kyber
- ▶ Baby-Kyber KEM
- ▶ Man in the Middle and Key Exchange Scenarios



## Un po' di storia

Come siamo arrivati qui

- 1994: Peter Shor pubblica un algoritmo ([Sho96; Sho94]) in grado di rompere RSA e altri schemi fattorizzando in primi in un modo impensabile per i computer classici
- 2005: Oded Regev pubblica una ricerca ([Reg05]) sul problema del Learning With Errors
- 2016: Il NIST richiede nuovi standard in grado di resistere all'avvento del quantum computing
- 2017: Viene proposto CRYSTALS-Kyber (KEM) ([Uff])
- 2018: Oded Regev vince il Premio Gödel per la sua ricerca
- 2021: CRYSTALS-Kyber è l'unico KEM reso standard ([Nis])
- Esso è ora usato in varie applicazioni tra cui Signal, WhatsApp, un branch di OpenSSL e altri.



# Algoritmo di Shor

Fattorizzazione quantistica

- La sicurezza di molti sistemi di crittografia asimmetrica o di scambio di chiavi si basa sulla difficoltà della fattorizzazione
- La difficoltà con l'algoritmo classico migliore che ci sia noto è al momento di

$$O\left(e^{1.9(\log N)^{1/3}(\log \log N)^{2/3}}\right)$$

- L'algoritmo di Shor ([Sho96; Sho94]), pubblicato nel 1994, è in grado di fattorizzare un numero in solo

$$O((\log N)^2(\log \log N))$$

- Per ora non ancora applicabile su grandi numeri
- I computer quantistici continuano a migliorare ([ST21])



# Lattice-based Cryptography

Cosa è un lattice

## Lattice

Un lattice è, in geometria e nella teoria dei gruppi, un insieme infinito di punti in uno spazio vettoriale tale che la somma o sottrazione delle coordinate di due punti nel lattice produce le coordinate un terzo punto, sempre appartenente al lattice.

Alcune costruzioni basate sui lattici ([Svp]) sembrano, al momento, resistenti ad attacchi ([Reg05]) da parte di calcolatori non solo classici ma anche quantistici!



# Lattice-based Cryptography

Learning With Errors

## Learning with Errors

Il problema del Learning with Errors (LWE) ([Svp]) è un problema matematico basato sull'idea di rappresentare le informazioni come un insieme di equazioni aggiungendo del rumore, ossia degli errori.

Esso è stato introdotto da Oded Regev nel suo lavoro del 2005 ([Reg05]) ed è stato dimostrato avere la stessa complessità di molti problemi worst-case aventi a che fare con i lattici.



# Table of Contents

## 2 CRYSTALS-Kyber

- ▶ Key Exchange and Quantum Security
- ▶ **CRYSTALS-Kyber**
- ▶ Baby-Kyber KEM
- ▶ Man in the Middle and Key Exchange Scenarios





# Public Key Encryption Scheme

2 CRYSTALS-Kyber

Per avvicinarsi al KEM occorre costruire delle primitive da poter usare in seguito. Queste sono quelle tipiche della crittografia a chiave pubblica ([Jop17]) ossia

- Key Generation
- Encryption
- Decryption



# Key Generation

## 2 CRYSTALS-Kyber

La generazione delle chiavi avviene nel seguente modo:

Kyber.CPA.KeyGen():

```
1  $\rho, \sigma \leftarrow \{0, 1\}^{256}$   
2  $\mathbf{A} \sim R_q^{k \times k} := \text{Sam}(\rho)$   
3  $(\mathbf{s}, \mathbf{e}) \sim \beta_\eta^k \times \beta_\eta^k := \text{Sam}(\sigma)$   
4  $\mathbf{t} := \text{Compress}_q(\mathbf{A}\mathbf{s} + \mathbf{e}, d_t)$   
5  $\text{return}(pk := (\mathbf{t}, \rho), sk := \mathbf{s})$ 
```

- prendo due valori casuali  $\rho, \sigma$  da 256 bit
- genero una matrice  $\mathbf{A}$  di dimensioni  $k \times k$  con coefficienti modulo  $q$  a partire da  $\rho$
- ottengo i valori di chiave privata  $\mathbf{s}$  ed errore  $\mathbf{e}$  come array di dimensione  $k$  con coefficienti piccoli (modulo  $\eta$ ) a partire da  $\sigma$
- ottengo  $\mathbf{t}$  come compressione di  $\mathbf{A}\mathbf{s} + \mathbf{e}$
- restituisco la coppia chiave pubblica, chiave privata



# Encryption

## Parte A

Kyber.CPA.Enc(pk,m):

```
1  $r \leftarrow \{0, 1\}^{256}$   
2  $\mathbf{t} := \text{Decompress}_q(\mathbf{t}, d_t)$   
3  $\mathbf{A} \sim R_q^{k \times k} := \text{Sam}(\rho)$   
4  $(\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta := \text{Sam}(\mathbf{r})$   
5  $\dots$ 
```

- creo un valore casuale di inizializzazione  $r$
- ottengo  $\mathbf{t}$  dal parametro compresso ottenuto in input ( $pk = (\mathbf{t}, \rho)$ )
- mi ricostruisco la matrice  $\mathbf{A}$  usando una Extendable Output Function a partire sempre da  $\rho$  (si ricorda che le funzioni Sam sono deterministiche)
- ottengo  $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2$  come due array di polinomi e un polinomio con coefficienti *piccoli*



# Encryption

## Parte B

Kyber.CPA.Enc(pk,m):

```
4  ...  
5   $\mathbf{u} := \text{Compress}_q(\mathbf{A}^T \mathbf{r} + \mathbf{e}_1, d_u)$   
6   $\mathbf{v} := \text{Compress}_q(\mathbf{t}^T \mathbf{r} + e_2 + \lceil \frac{q}{2} \rceil \cdot m, d_v)$   
7  return  $c := (\mathbf{u}, \mathbf{v})$ 
```

- calcolo  $\mathbf{u}$  usando gli errori vettoriali generati precedentemente
- calcolo  $\mathbf{v}$  comprimendo il risultato della somma di rumore da due fonti  $(\mathbf{t}^T \mathbf{r}, e_2)$  e il messaggio appositamente amplificato in modo che abbia coefficienti grandi
- restituisco il ciphertext, ossia la coppia  $(\mathbf{u}, \mathbf{v})$



# Decryption

## 2 CRYSTALS-Kyber

Kyber.CPA.Dec(sk,c):

```
1  $\mathbf{u} := \text{Decompress}_q(\mathbf{u}, d_u)$   
2  $\mathbf{v} := \text{Decompress}_q(\mathbf{v}, d_v)$   
3 return  $\text{Compress}_q(\mathbf{v} - \mathbf{s}^T \mathbf{u}, 1)$ 
```

- ottengo  $\mathbf{u}$  e  $\mathbf{v}$  decomprimendo il ciphertext
- ricavo il plaintext come  $\mathbf{v} - \mathbf{s}^T \mathbf{u}$  e approssimando: se il valore del coefficiente è più vicino a 0 o  $q$  che a  $\lceil \frac{q}{2} \rceil$  allora avrò uno 0, altrimenti un 1



# Messaggi e polinomi

2 CRYSTALS-Kyber

Noi siamo abituati a trasmettere i messaggi nella forma di numeri, tuttavia qui lavoriamo con i polinomi. Come è possibile questo?

## I numeri sono polinomi

Ci basta effettuare una semplice conversione prendendo il messaggio in binario e associando ad ogni bit un esponente, ad esempio  $11_{10} = 1011_2 = x^3 + x + 1$  ([Jop17; Bab]).

## Amplificazione del polinomio

Per evitare che i rumori introdotti come parte fondamentale della crittografia vadano a sovrascrivere il nostro messaggio rendendolo inintelligibile amplifichiamo il nostro messaggio di una quantità pari alla metà di  $q$ , così lo riusciamo a distinguere ([Jop17; Bab]). Questo risulta evidente dallo pseudocodice dell'encryption.



# Correttezza di Kyber.CPA

2 CRYSTALS-Kyber

## Correttezza

Kyber.CPA è  $(1 - \delta)$ –corretto con un  $\delta < 2^{-128}$  ([Jop17]). Questo significa che una piccola quantità di decryption potrebbero non portare al valore realmente desiderato. Questa quantità è però trascurabile.



# Costruire un KEM

## Premessa

Per costruire un KEM dovremo usare le primitive di Key Exchange costruite prima per ottenere due funzionalità:

- Encaps: questa funzione ci fornirà la chiave da proporre e un ciphertext avendo a disposizione una chiave pubblica
- Decaps: questa funzione prende in input una chiave privata e un ciphertext e ottiene la chiave concordata

Esso ovviamente necessita anche di KeyGen, che tuttavia rimane identico a prima ([Jop17])





# Encaps

## 2 CRYSTALS-Kyber

Kyber.Encaps(pk):

```
1  $m \leftarrow \{0, 1\}^{256}$   
2  $(\hat{K}, r) := G(H(pk), m)$   
3  $(\mathbf{u}, \mathbf{v}) := \text{Kyber.CPA.Enc}((\mathbf{t}, \rho), m; r)$   
4  $\mathbf{c} := (\mathbf{u}, \mathbf{v})$   
5  $K := H(\hat{K}, H(\mathbf{c}))$   
6 return  $(\mathbf{c}, K)$ 
```

- genero un valore casuale  $m$
- ottengo il seme dei numeri casuali e parte della chiave dagli hash di  $m$
- cifro il valore  $m$  usando  $r$  come seme
- uso sia il ciphertext che il plaintext per ottenere la chiave
- trasmetto il ciphertext



# Decaps

## Parte A

Kyber.Decaps(sk z c):

```
1  $m' := \text{Kyber.CPA.Dec}(s, (\mathbf{u}, \mathbf{v}))$   
2  $(\hat{K}', r') := G(H(pk), m')$   
3  $(\mathbf{u}', \mathbf{v}') := \text{Kyber.CPA.Enc}((\mathbf{t}, \rho), m'; r')$   
4  $\dots$ 
```

- decifro il messaggio che mi è arrivato
- ottengo il seed e la  $\hat{K}'$
- provo a cifrare nuovamente con gli stessi parametri (ho eliminato il non determinismo della cifratura)



# Decaps

## Parte B

Kyber.Decaps(sk z c):

```
3  ...  
4  if  $(\mathbf{u}', v') == (\mathbf{u}, v)$  then  
5      return  $K := H(\hat{K}', H(c))$   
6  else  
7      return  $K := H(z, H(c))$   
8  end if
```

- se il ciphertext coincide allora restituisco la chiave
- altrimenti restituisco una chiave sbagliata generata a partire da  $z$



# Funzioni di Hashing

2 CRYSTALS-Kyber

Per eseguire alcune operazioni nell'incapsulamento e decapsulamento si fa uso di funzioni di hashing. Seguendo l'articolo [Jop17] noi useremo due varianti Keccak, ossia `sha3-256` come  $H$  e `sha3-512` come  $G$ , il cui output sarà trasformato in polinomi seguendo le solite regole.



# Table of Contents

## 3 Baby-Kyber KEM

- ▶ Key Exchange and Quantum Security
- ▶ CRYSTALS-Kyber
- ▶ **Baby-Kyber KEM**
- ▶ Man in the Middle and Key Exchange Scenarios



## Una versione ridotta

Un esempio concreto e una implementazione

### Avviso

Per vedere un esempio ridotto fare riferimento a [Bab]. L'uso di parametri così bassi potrebbe aumentare il tasso di errori. Inoltre, questo sistema non fa uso di compressione, a differenza del vero CRYSTALS-Kyber [Jop17; Uff].

Qui non sarà riportata questa versione quanto una rappresentazione del protocollo specifico realizzato come componente chiave del sistema di chat LightKnife.



# Table of Contents

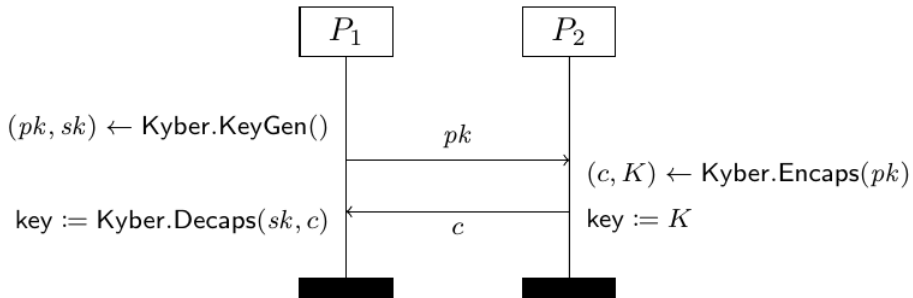
## 4 Man in the Middle and Key Exchange Scenarios

- ▶ Key Exchange and Quantum Security
- ▶ CRYSTALS-Kyber
- ▶ Baby-Kyber KEM
- ▶ Man in the Middle and Key Exchange Scenarios



# Kyber.KE protocol

Logical view

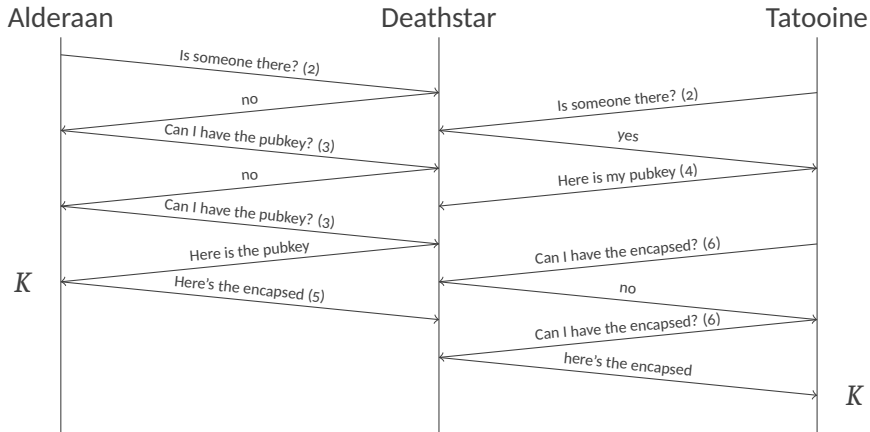






# Kyber.KE protocol

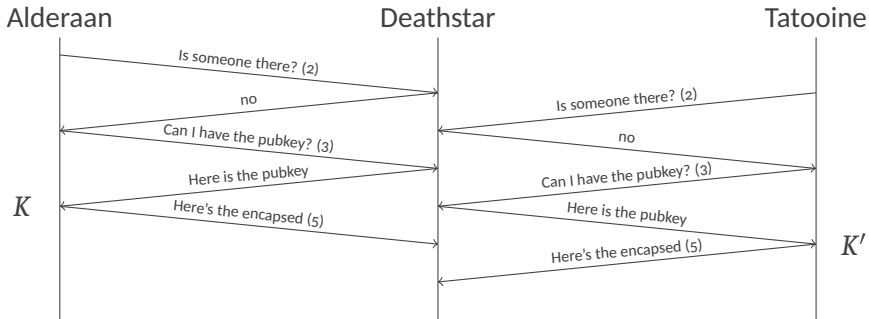
Network view





# Man in the Middle against Kyber.KE

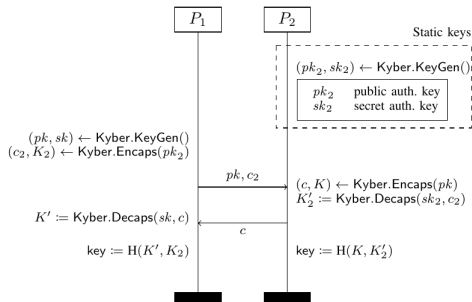
## 4 Man in the Middle and Key Exchange Scenarios



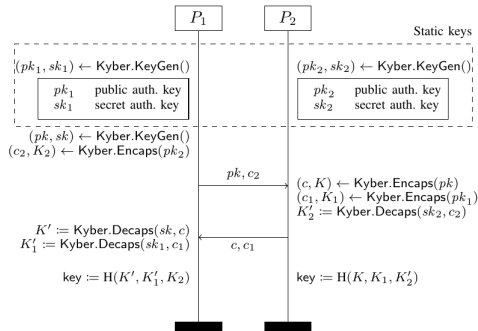


# Kyber.UAKE and Kyber.AKE

## 4 Man in the Middle and Key Exchange Scenarios



One-side Authenticated Key Exchange  
[Jop17]



Authenticated Key Exchange



# Post-Quantum Cryptography Standards

*Thank you for listening!*  
*Any questions?*



## Bibliografia

### 5 Man in the Middle and Key Exchange Scenarios

- [Bab] *Kyber - How does it work?* URL: <https://cryptopedia.dev/posts/kyber/>. (accessed: 08.05.2024).
- [Jop17] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, Damien Stehlé. “CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM”. In: (2017).
- [Nis] *NIST Announces First Four Quantum-Resistant Cryptographic Algorithms*. URL: <https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms>. (accessed: 08.05.2024).



## Bibliografia

### 5 Man in the Middle and Key Exchange Scenarios

- [Reg05] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing* (2005). URL: <https://dl.acm.org/doi/10.1145/1060590.1060603>.
- [Sho94] Peter W. Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994). URL: <https://ieeexplore.ieee.org/document/365700>.
- [Sho96] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (1996). URL: <https://arxiv.org/abs/quant-ph/9508027>.



## Bibliografia

### 5 Man in the Middle and Key Exchange Scenarios

- [ST21] Unathi Skosana and Mark Tame. “Demonstration of Shor’s factoring algorithm for  $N=21$  on IBM quantum processors”. In: *Sci Rep* (2021). URL: <https://www.nature.com/articles/s41598-021-95973-w>.
- [Svp] *Learning with errors*. URL: [https://en.wikipedia.org/wiki/Learning\\_with\\_errors](https://en.wikipedia.org/wiki/Learning_with_errors). (accessed: 08.05.2024).
- [Uff] *CRYSTALS Cryptographic Suite for Algebraic Lattices*. URL: <https://pq-crystals.org/kyber/index.shtml>. (accessed: 08.05.2024).