Post-Quantum Cryptography Standards

Spiegazione e Implementazione del KEM CRYSTALS-Kyber

Sicurezza dei Sistemi e delle Reti Informatiche

Samuele Manclossi (09882A)

July 15, 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

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

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

- 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!

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

Public Key Encryption Scheme

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():

$\rho, \sigma \leftarrow \{0,1\}^{256}$
$\boldsymbol{A} \sim R_q^{k \times k} := \texttt{Sam}(\rho)$
$(\boldsymbol{s}, \boldsymbol{e}) \sim \beta_\eta^k \times \beta_\eta^k := \texttt{Sam}(\sigma)$
$\boldsymbol{t} := \texttt{Compress}_q(\boldsymbol{As} + \boldsymbol{e}, d_t)$
$return(pk := (\boldsymbol{t}, \rho), sk := \boldsymbol{s})$

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

Encryption

Parte A

Kyber.CPA.Enc(pk,m):

$$
\begin{aligned}
&\boldsymbol{r} \leftarrow \{0,1\}^{256}\\
&\boldsymbol{t} := \texttt{Decompress}_q(\boldsymbol{t}, d_t)\\
&\boldsymbol{A} \sim R_q^{k\times k} := \texttt{Sam}(\rho)\\
&(\boldsymbol{r}, \boldsymbol{e_1}, e_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta := \texttt{Sam}(\boldsymbol{r})\\
&\dots
\end{aligned}
$$

- creo un valore casuale di inizializzazione $\boldsymbol{r}$
- ottengo $\boldsymbol{t}$ dal parametro compresso ottenuto in input ($pk = (\boldsymbol{t}, \rho)$)
- mi ricostruisco la matrice $\boldsymbol{A}$ usando una Extendable Output Function a partire sempre da ρ (si ricorda che le funzioni Sam sono deterministiche)
- ottengo $\boldsymbol{r}, \boldsymbol{e_2}, e_2$ come due array di polinomi e un polinomio con coefficienti *piccoli*

Encryption

Parte B

Kyber.CPA.Enc(pk,m):

$\ldots$

$\boldsymbol{u} := \texttt{Compress}_q(\boldsymbol{A}^T\boldsymbol{r} + \boldsymbol{e_1}, d_u)$

$v := \texttt{Compress}_q(\boldsymbol{t}^T\boldsymbol{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$

$\texttt{return}\ c := (\boldsymbol{u}, v)$

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

Decryption

Kyber.CPA.Dec(sk,c):

$$
\begin{aligned}
&\boldsymbol{u} := \texttt{Decompress}_q(\boldsymbol{u}, d_u) \\
&v := \texttt{Decompress}_q(v, d_v) \\
&\texttt{return}\ \ \texttt{Compress}_q(v - \boldsymbol{s}^T\boldsymbol{u}, 1)
\end{aligned}
$$

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

Messaggi e polinomi

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 inintellegibile 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

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 cifratura 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

Kyber.Encaps(pk):

$$m \leftarrow \{0,1\}^{256}$$
$$(\hat{K}, r) := G(H(pk), m)$$
$$(\boldsymbol{u}, v) := \texttt{Kyber.CPA.Enc}((\boldsymbol{t}, \rho), m; r)$$
$$c := (\boldsymbol{u}, v)$$
$$K := H(\hat{K}, H(c))$$
$$\texttt{return}\ (c, K)$$

- 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):

$m' := \texttt{Kyber.CPA.Dec}(s, (\boldsymbol{u}, v))$
$(\hat{K}', r') := G(H(pk), m')$
$(\boldsymbol{u}', v') := \texttt{Kyber.CPA.Enc}((\boldsymbol{t}, \rho), m'; r')$
$\ldots$

- 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):

```
...
if (u', v') == (u, v) then
    return K := H(K̂', H(c))
else
    return K := H(z, H(c))
end if
```

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

Funzioni di Hashing

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

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].

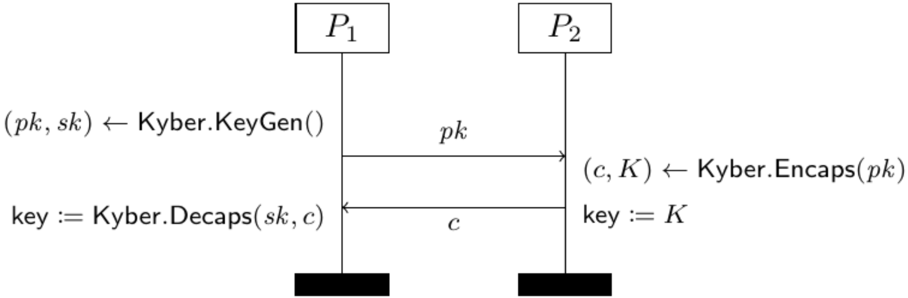
Qui non sarà riportata questa versione quanto una rappresentazione del protocollo specifico realizzato come componente chiave del sistema di chat LightKnife.

Table of Contents

4 Man in the Middle and Key Exchange Scenarios

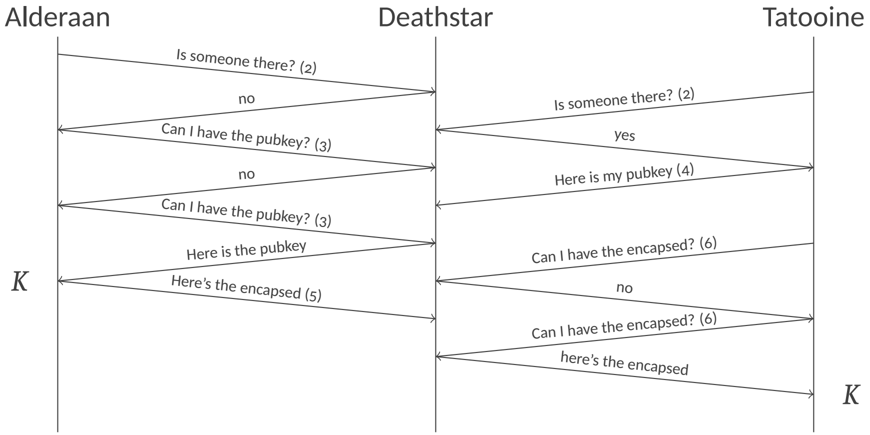
Kyber.KE protocol

Logical view



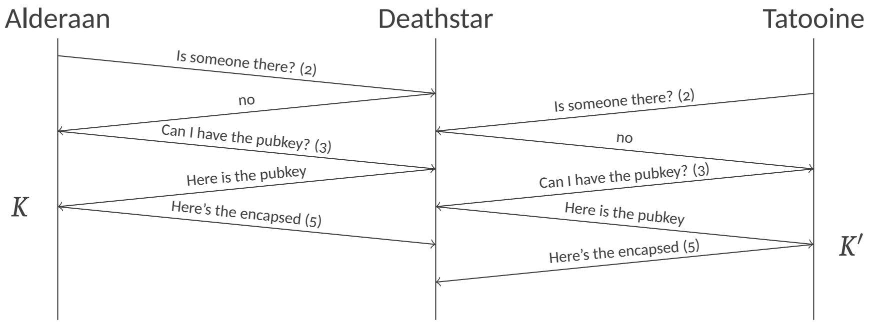
Kyber.KE protocol

Network view



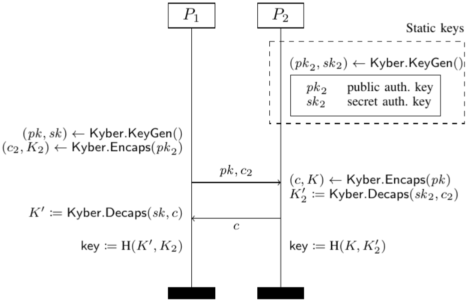
Man in the Middle against Kyber.KE

4 Man in the Middle and Key Exchange Scenarios

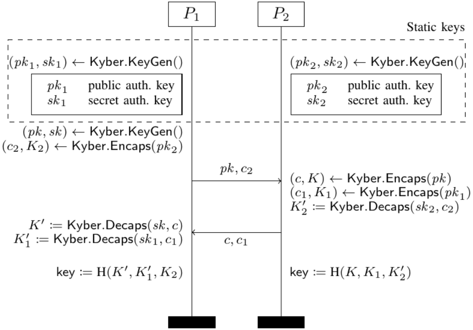


Kyber.UAKE and Kyber.AKE

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 , Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanckk, 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

[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

[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. (accessed: 08.05.2024).

[Uff] *CRYSTALS Cryptographic Suite for Algebraic Lattices*. URL: https://pq-crystals.org/kyber/index.shtml. (accessed: 08.05.2024).