ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea in Ingegneria e Scienze Informatiche

Post-Quantum Cryptography

Tesi di laurea in:
Crittografia

Relatore
Prof. Margara Luciano

Candidato
Olivieri Michele

III Sessione di Laurea
Anno Accademico 2024-2025

Abstract

La sicurezza digiatle moderna pone le sue fondamenta sulla crittografia a chiave pubblica, che a sua volta delega la sua robustezza alla difficoltà computazione di problemi matematici come il problema della fattorizzazione e del logaritmo discreto. Tuttavia, l'avvento dei computer quantistici minaccia di compromettere la sicurezza di questi sistemi, poiché algoritmi quantistici come quello di Shor possono risolvere tali problemi in tempo polinomiale sfruttando la trasformata di Fourier quantistica per individuare la periodicità di funzioni modulari. Sebbe i comuper quantistici attuali non siano ancora in grado di violare chiavi crittografiche reali, il modello di minaccia "harvest now, decrypt later" rende la transizione urgente già oggi.

In risposta, la crittografia post-quantistica sviluppa algoritmmi fondati su famiglie di problemi matematici considerati resistenti anche ad attacchi quantistici: reticoli algebrici, codici correttori di errori e funzioni hash. Il NIST ha pubblicato nel 2024 i primi standar ufficiali ML-KEM, ML-DSA e SLH-DSA, segnando l'inizio concreto della transizione. Questa tesi analizza i fondamenti teorici di questa evoluzione, dall'impatto di Shor sulla crittografia classica alle soluzioni post-quantistiche, evidenziando come la migrazione verso questi nuovi paradigmi rappresenti una delle sfide più attuali e urgenti per la sicurezza informatica globale del prossimo decennio.

The enemy knows the system.
Claude E. Shannon

Contents

List of Figures

Chapter 1

Introduction

Contesto e motivazioni Per comprendere l'importanza della crittografia post-quantistica, è fondamentale analizzare il contesto in cui essa si inserisce e le motivazioni che ne hanno guidato lo sviluppo.

La crittografia classica, come visto a lezione, pone le sue fondamenta su problemi computazionalmente difficili per i quali non esistono algoritmi efficienti in grado di risolverli in tempi polinomiali.

Tuttavia, l'emergere dei computer quantistici ha introdotto una nuova dimensione nel panorama della sicurezza informatica. Questi dispositivi sfruttando i principi della meccanica quantistica sono in grado di eseguire calcoli in modo radicalmente diverso rispetto ai computer classici, rendendoli potenzialmente capaci di risolvere in tempi polinomiali quei problemi matematici che risultano intrattabili per le macchine convenzionali [NC10].

Minaccia dei computer quantistici: Perché è necessaria? L'ipotesi di una minaccia quantistica emerge già nel ventesimo secolo, quando nel 1994 l'algoritmo di Shor [Sho94] dimostra che un computer quantistico sufficientemente potente potrebbe fattorizzare grandi numeri interi e calcolare logaritmi discreti in tempo polinomiale. Questo rende vulnerabili algoritmi come RSA, ECC e DSA, che costituiscono la base della sicurezza informatica moderna.

Per molti anni il problema è rimasto solo teorico, perché il calcolo quantistico non aveva applicazioni pratiche. Lo scenario è cambiato con i recenti progressi tecnologici e lo sviluppo dei primi prototipi avanzati da parte di leader industriali

come Google e Microsoft, che hanno riportato risultati significativi con i loro progetti: Willow[1] di Google e il processore Maiorana 1[2] di Microsoft. Questi progetti hanno quindi portato alla luce dei veri calcolatori quantistici funzionanti e nonostante il limitato numero di qbit stabili sia insufficiente per applicazioni pratiche su larga scala, questo segna una svolta nel settore.

Sebbene le macchine quantistiche siano ancora in fase sperimentale, diversi scienziati ritengono che la loro costruzione su larga scala sia ormai una sfida principalmente ingegneristica, e alcuni prevedono la loro maturazione entro i prossimi vent'anni.

Considerando che l'attuale infrastruttura crittografica ha richiesto quasi due decenni per essere implementata, risulta necessario iniziare da subito la transizione verso sistemi progettati per resistere al calcolo quantistico [Ber09].

Obiettivi della crittografia post-quantistica L'obiettivo della crittografia post-quantistica è quindi sviluppare algoritmi crittografici sicuri sia contro i computer quantistici che classici, garantendo così un futuro sicuro anche in un'era dominata dai computer quantistici.

La crittografia post-quantistica non si limita a sostituire gli algoritmi vulnerabili, ma mira a costruire un'infrastruttura di sicurezza robusta e duratura, capace di adattarsi alle sfide tecnologiche future mantenendo la compatibilità con i protocolli e le reti esistenti[3].

Structure of the Thesis Lo scopo della relazione invece è quindi quello di fornire una panoramica completa, introducendo in primo luogo i fondamenti della crittografia classica, per capire dove risiedono le vulnerabilità che i computer quantistici sono in grado di sfruttare. Una volta capiti tali principi, procederemo ad analizzare Shor, il suo impatto su tali algoritmi ed infine le soluzioni proposte dalla crittografia post-quantistica.

[1] Google Quantum AI, "Willow: A quantum computing milestone", Dicembre 2024. Disponibile su: blog.google

[2] Microsoft Quantum, "Maiorana 1: The first milestone on the path to a quantum supercomputer", Ottobre 2024. Disponibile su: www.microsoft.com

[3] NIST, "Post-Quantum Cryptography Standardization", `https://csrc.nist.gov/projects/post-quantum-cryptography`

Chapter 2

Fondamenti di crittografia classica

Come anticipato, per comprendere le sfide poste dai computer quantistici alla crittografia moderna, è fondamentale conoscerne i principi di base dietro i principali algoritmi crittografici attualmente in uso, in particolar modo ci concentreremo sui protocolli a chiave pubblica.

2.1 Crittografia a chiave pubblica: RSA, ECC

La crittografia a chiave pubblica, introdotta nel 1976 da Diffie, Hellman e Merkle [DH76], costituisce una svolta fondamentale nel panorama della sicurezza informatica moderna. A differenza dei sistemi simmetrici, che vincolano mittente e destinatario alla condivisione di un unico segreto, i protocolli asimmetrici impiegano una coppia di chiavi: una pubblica k_{pub}, liberamente distribuibile, e una privata k_{prv}, mantenuta segreta dal proprietario. La sicurezza di questo sistema si basa sulla difficoltà di risalire alla chiave privata a partire da quella pubblica. Le funzioni di cifratura C e decifratura D sono note a tutti, e per ogni messaggio m deve valere:

$$D(C(m; k_{\text{pub}}); k_{\text{prv}}) = m.$$

Il funzionamento di questo sistema si basa sulle funzioni one-way trapdoor: operazioni matematiche semplici da eseguire in una direzione, ma computazionalmente intrattabili da invertire senza la conoscenza di una informazione specifica (la "trappola").

La teoria dei numeri e l'algebra modulare forniscono il substrato matematico necessario per generare tali funzioni; a seconda del problema matematico sottostante, si distinguono i vari algoritmi di crittografia asimmetrica oggi in uso.

Richiami di algebra modulare L'aritmetica modulare è un sistema in cui i numeri si riavvolgono entro un intervallo fissato da un modulo n. Quando un valore supera (o scende sotto) questo intervallo, viene riportato all'interno prendendo il resto della divisione per n.

Un esempio quotidiano è l'orologio: in un sistema a 12 ore il modulo è 12. Se sono le 10 e aggiungo 4 ore, il risultato non è 14, ma 2, perché:

$$14 \equiv 2 \pmod{12}.$$

Per calcolare $c = a \bmod b$ si considera il resto della divisione intera tra a e b, ottenendo un valore sempre compreso tra 0 e $b - 1$, esempio: $6 \bmod 4 = 2$.

Problemi difficili

- **Fattorizzazione:** dati p, q è facile calcolare $n = pq$; dato n è difficile trovare p e q.
- **Radice modulare:** dato $y = x^z \bmod s$ invertire la potenza è difficile senza conoscere $\varphi(s)$.
- **Logaritmo discreto:** data $y = x^z \bmod s$ trovare z è computazionalmente difficile.

2.1.1 RSA

Il cifrario RSA, proposto da Rivest, Shamir e Adleman nel 1978 [RSA78], è il sistema crittografico a chiave pubblica più diffuso e studiato. La sua sicurezza si fonda sulla difficoltà computazionale della fattorizzazione di numeri interi molto grandi[1].

[1] Materiale didattico del corso di Crittografia, Università di Bologna, a.a. 2024/2025

Generazione delle chiavi Ogni utente genera la propria coppia di chiavi attraverso i seguenti passaggi:

1. Scelta di due numeri primi p e q molto grandi
2. Calcolo di $n = pq$ e della funzione di Eulero $\phi(n) = (p-1)(q-1)$
3. Scelta di un intero e minore di $\phi(n)$ e coprimo con esso
4. Calcolo dell'intero d, inverso moltiplicativo di e modulo $\phi(n)$

La chiave pubblica è la coppia (e, n), mentre la chiave privata è d. La cifratura di un messaggio m avviene calcolando $c = m^e \bmod n$, mentre la decifratura richiede il calcolo di $m = c^d \bmod n$.

La correttezza dell'algoritmo è garantita dal teorema di Eulero: poiché $ed \equiv 1 \pmod{\phi(n)}$, si ha $ed = 1 + k\phi(n)$ per qualche intero k, e quindi:

$$m^{ed} \bmod n = m^{1+k\phi(n)} \bmod n = m \cdot (m^{\phi(n)})^k \bmod n = m \bmod n$$

Sicurezza e dimensioni delle chiavi La sicurezza di RSA dipende da l'impossibilità pratica di fattorizzare n quando questo è sufficientemente grande. Conoscendo la fattorizzazione $n = pq$, un attaccante potrebbe infatti calcolare $\phi(n)$ e di conseguenza la chiave privata d.

Attualmente, le dimensioni delle chiavi considerate sicure sono di almeno 2048 bit, con raccomandazioni crescenti verso 4096 bit per applicazioni che richiedono sicurezza a lungo termine [Nat20]. Chiavi di 1024 bit sono considerate obsolete e vulnerabili ad attacchi con risorse computazionali moderne.

2.1.2 Crittografia su Curve Ellittiche (ECC)

La crittografia su curve ellittiche, sviluppata indipendentemente da Neal Koblitz e Victor Miller nel 1985 [Kob87, Mil85], offre un'alternativa matematicamente elegante e computazionalmente efficiente a RSA.

Fondamenti matematici Una curva ellittica su un campo finito $\mathbb{Z}_p$ (con p primo e $p > 3$) è definita dall'equazione di Weierstrass in forma normale:

$$y^2 = x^3 + ax + b$$

dove $a, b \in \mathbb{Z}_p$ soddisfano la condizione $4a^3 + 27b^2 \bmod p \neq 0$, che garantisce l'assenza di punti singolari sulla curva[2].

L'insieme dei punti (x, y) che soddisfano questa equazione, insieme al punto all'infinito $\mathcal{O}$, forma un gruppo abeliano additivo. È possibile definire un'operazione di addizione tra punti della curva tale che, dati due punti P e Q, la loro somma $P + Q$ sia ancora un punto della curva.

Per punti distinti $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$, con $P \neq -Q$, si ha:

$$\lambda = \frac{y_Q - y_P}{x_Q - x_P}, \quad x_S = \lambda^2 - x_P - x_Q, \quad y_S = -y_P + \lambda(x_P - x_S)$$

dove $S = P + Q = (x_S, y_S)$. Nel caso di raddoppio di un punto ($P = Q$), il coefficiente angolare diventa $\lambda = \frac{3x_P^2 + a}{2y_P}$.

Il problema del logaritmo discreto La sicurezza di ECC si basa sulla difficoltà del problema del logaritmo discreto su curve ellittiche (ECDLP): dati due punti P e Q sulla curva, trovare l'intero k tale che $Q = kP$, dove kP denota l'addizione di P con se stesso k volte.

La moltiplicazione scalare $Q = kP$ è computazionalmente efficiente (tempo polinomiale), mentre il problema inverso è considerato intrattabile: tutti gli algoritmi classici noti hanno complessità esponenziale nella dimensione della chiave.

[2]Materiale didattico del corso di Crittografia, Università di Bologna, a.a. 2024/2025

Vantaggi rispetto a RSA ECC offre lo stesso livello di sicurezza di RSA con chiavi significativamente più corte. Una chiave ECC di 256 bit fornisce una sicurezza paragonabile a una chiave RSA di 3072 bit [Nat20]. Questo si traduce in:

- Minore occupazione di memoria e larghezza di banda
- Operazioni crittografiche più veloci
- Minore consumo energetico, cruciale per dispositivi mobili e IoT

2.1.3 Sicurezza classica

Entrambi gli algoritmi sono considerati sicuri nell'ambito del calcolo classico per dimensioni di chiave appropriate. La loro robustezza deriva dalla complessità computazionale dei problemi matematici sottostanti:

- **Fattorizzazione per RSA**: il miglior algoritmo classico noto è il General Number Field Sieve (GNFS), con complessità sub-esponenziale [LL93]
- **ECDLP per ECC**: gli algoritmi più efficienti, come il metodo rho di Pollard, hanno complessità completamente esponenziale $O(\sqrt{n})$, dove n è l'ordine del gruppo [Pol78]

Questa differenza nella complessità degli attacchi spiega perché ECC richiede chiavi più corte per garantire lo stesso livello di sicurezza.

RSA e ECC costituiscono oggi la base dell'infrastruttura di sicurezza digitale globale, utilizzati in TLS/SSL per la sicurezza web, in SSH per l'accesso remoto sicuro, nella firma digitale di documenti e software, e in numerose altre applicazioni critiche.

2.2 Limiti rispetto ai computer quantistici

Avendo introdotto i fondamenti della crittografia classica possiamo provare ora ad analizzare le vulnerabilità di questi algoritmi in particolar modo rispetto al calcolo quantistico. Infatti nei prossimi capitoli capiremo quali sono i vantaggi che i computer quantistici offrono rispetto a quelli classici e come questi possono compromettere la sicurezza degli algoritmi classici. in particolare esamineremo l'algoritmo di Shor e il suo impatto su RSA.

Chapter 3

Computazione quantistica e impatto sulla crittografia classica

Entriamo ora nel vivo della relazione: sviscerare i segreti dietro l'algoritmo di Shor per capire in che modo rompere il protocllo RSA appena descritto, e analogamente per l'algoritmo di Grover per capire che impatto ha sulla crittografia simmetrica. Per farlo avremo prima bisogno di introdurre alcuni concetti di base della computazione quantistica tali algoritmi utilizzano per ottenere i loro risultati. Una volta compresi questi concetti potremo procedere alla spiegazione degli algoritmi.

3.1 Nozioni base di computazione quantistica

La computazione quantistica rappresenta un paradigma di calcolo che sfrutta i principi della meccanica quantistica per elaborare l'informazione in modi che non sono possibili con i computer classici[NC10, Sezione 1.1]. Di questi principi di meccanica quantistica non esiste un corrispettivo diretto nel modelli di calcolo classico, e proprio per questo motivo che ne analizziamo gli effetti e le conseguenze.

3.1.1 Richiami essenziali di meccanica quantistica

In questa sezione non si fornisce una trattazione formale della teoria, ma si introducono i concetti essenziali necessari a comprendere il funzionamento dei computer

quantistici e dei meccanismi che consentono loro di superare i limiti della computazione classica.

Il primo concetto fondamentale è quello di ***stato quantistico***. Mentre un bit classico può assumere esclusivamente i valori 0 o 1, un sistema quantistico può trovarsi in una *sovrapposizione* di stati[NC10, Sezione 1.2]. Questo non significa che sia 0 e 1 "contemporaneamente" in senso magico ma semplicemente che lo stato di un qubit può essere descritto come una combinazione lineare degli stati base $|0\rangle$ e $|1\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

con coefficienti complessi che ne determinano le probabilità di osservazione[NC10, p. 13].

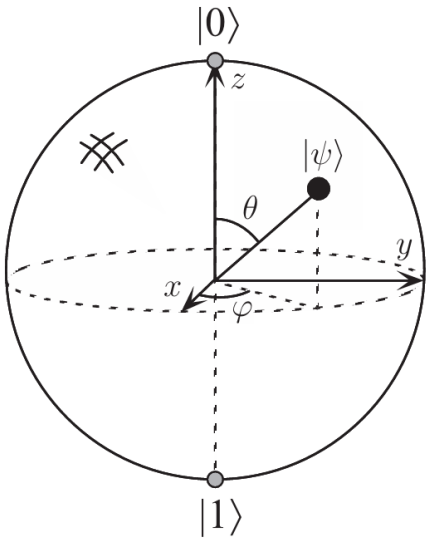


Figure 3.1: Bloch sphere representation of a qubit.

Un secondo concetto chiave è l'***entanglement***, una forma di correlazione quantistica tra più qubit per cui lo stato complessivo del sistema non può essere descritto come il prodotto degli stati dei singoli qubit[NC10, Sezione 2.2.8]. In presenza di entanglement, la misura di un qubit influisce istantaneamente sullo stato degli altri, indipendentemente dalla distanza che li separa[NC10, Sezione 1.2.1]. Questa proprietà consente di rappresentare e manipolare informazioni in modo non separabile, ciò significa che le operazioni su un qubit possono avere effetti

immediati sugli altri qubit entangled con esso.

Il terzo elemento fondamentale è la ***misura***. Quando uno stato quantistico viene misurato, la sovrapposizione collassa e il risultato ottenuto è un valore classico[NC10, Sezione 1.2]. L'esito della misura è probabilistico e dipende dallo stato del sistema immediatamente prima dell'osservazione[NC10, Sezione 2.2.3].

Sovrapposizione, entanglement e misura costituiscono i principi alla base della differenza concettuale tra computazione classica e computazione quantistica, e determinano il modo in cui l'informazione viene elaborata in un sistema quantistico[NC10, Sezione 1.3].

Qubit e operazioni quantistiche

L'unità fondamentale di informazione in un computer quantistico è quindi il qubit[NC10, Sezione 1.2]. Dal punto di vista matematico, un qubit è rappresentato come un vettore unitario in uno spazio di Hilbert bidimensionale, mentre un registro composto da n qubit è descritto in uno spazio di dimensione 2^n.

Le operazioni sui qubit sono realizzate mediante *porte quantistiche*, che corrispondono a trasformazioni lineari e reversibili, descritte da operatori unitari. A differenza delle porte logiche classiche, le porte quantistiche agiscono su stati in sovrapposizione, modificando simultaneamente tutte le componenti dello stato quantistico.

Matematicamente, queste porte sono rappresentate come matrici unitarie, il che conferisce loro due proprietà cruciali: la somma delle probabilità dei possibili stati deve essere sempre pari a 1 e ogni operazione deve essere reversibile (ovvero, ogni porta ha una sua inversa che può annullare l'operazione). Le principali porte quantistiche[NC10, Sezioni 1.3.1, 1.3.2, 1.3.4, 1.3.6, 4.2, 4.5, 5.1]:

- porte a bit singolo come: la porta Hadamard (H), che viene usata per mettere i qubit in superposizioni, la porta Pauli-X (equivalente a una NOT classica), e le porte di Pauli(X, Y, Z) che ruotano il vettore.

- porte a più bit che realizzano operazioni condizionate tra due qubit, fondamentali per creare entanglement.

- porta SWAP che scambia lo stato di due qubit, la porta di Fase Controllata (Rk) utile per la QFT.

- porte specializzate costruite per specifici algoritmi a partire da queste.

L'uso combinato di queste porte consente di elaborare, in un singolo passo computazionale, un insieme di stati che cresce esponenzialmente con il numero di qubit. Tale fenomeno è noto come *parallelismo quantistico* e rappresenta una delle principali fonti di vantaggio rispetto al calcolo classico, pur non traducendosi automaticamente in un'accelerazione per qualsiasi problema[NC10, Sezioni 1.4.2, 4.5.4].

Modello di calcolo quantistico

Il funzionamento di un computer quantistico segue un modello di calcolo specifico, distinto da quello deterministico utilizzato nei computer classici. Un algoritmo quantistico è generalmente articolato in tre fasi principali[NC10, Sezioni 1.3.4, 4.6]:

- preparazione dello stato iniziale;

- applicazione di una sequenza di porte quantistiche;

- misura finale del sistema.

Durante la fase di evoluzione, lo stato quantistico del sistema viene trasformato in modo deterministico secondo le leggi della meccanica quantistica. La natura probabilistica emerge esclusivamente al momento della misura, quando lo stato viene proiettato su uno dei possibili risultati osservabili.

La potenza del modello di calcolo quantistico non risiede nella possibilità di valutare esplicitamente tutte le soluzioni di un problema, bensì nella capacità di sfruttare il fenomeno dell'*interferenza quantistica*. Attraverso opportune sequenze di operazioni, le ampiezze associate alle soluzioni corrette possono essere amplificate, mentre quelle delle soluzioni errate vengono attenuate[NC10, Sezioni 1.4.3, 6.1.3].

3.1.2 Collegamento con la crittografia

I concetti introdotti in questa sezione costituiscono il fondamento teorico per analizzare l'impatto della computazione quantistica sulla crittografia moderna. In particolare, l'uso combinato di sovrapposizione, entanglement e interferenza consente di affrontare in modo efficiente problemi come la fattorizzazione di interi e il calcolo del logaritmo discreto[NC10, Sezioni 1.4, 5.3].

3.2 Algoritmi quantistici rilevanti: Shor

L'algoritmo di Shor rappresenta uno dei risultati più significativi nel campo della computazione quantistica, non solo per le sue implicazioni sulla crittografia, ma anche per la profondità delle idee matematiche che lo compongono[NC10, Sezione 5.3]. Infatti per comprendere appieno il funzionamento di questo algoritmo, è necessario procedere con un'analisi rigorosa che parta dai fondamenti matematici, prescindendo inizialmente dagli aspetti quantistici.

L'obiettivo quindi di questa sezione è chiarire perché l'algoritmo funziona dal punto di vista matematico, e solo in una fase successiva dell'analisi vedremo come viene accelerato dal computer quantistico.

Il problema della fattorizzazione

Il punto di partenza è il problema della fattorizzazione di interi. Dato un intero composto $N = p \cdot q$, dove p e q sono numeri primi di grandi dimensioni, il problema consiste nel determinare p e q conoscendo solamente N. La difficoltà computazionale della fattorizzazione deriva da due caratteristiche fondamentali: in primo luogo, non è noto alcun algoritmo classico in grado di fattorizzare un numero in tempo polinomiale rispetto alla dimensione dell'input; in secondo luogo, i tentativi diretti di fattorizzazione crescono rapidamente con l'aumentare della dimensione di N, rendendo il problema intrattabile per valori sufficientemente grandi.

L'intuizione fondamentale di Shor consiste nel trasformare il problema della fattorizzazione in un problema di natura diversa, caratterizzato da una struttura matematica più favorevole[NC10, Sezione 5.3.2]. Questa trasformazione permette di ricondurre la ricerca dei fattori primi a un problema di teoria dei numeri che,

come vedremo, può essere risolto in modo efficiente sfruttando le peculiarità della computazione quantistica.

3.2.1 Dalla fattorizzazione alla teoria dei numeri

La strategia di Shor si basa sull'osservazione che la fattorizzazione può essere ricondotta al problema della determinazione del periodo di una funzione. Questa connessione non è immediata e richiede una serie di passaggi matematici che è necessario esplicitare con precisione.

Il primo passo consiste nella scelta di un numero intero a tale che $1 < a < N$ e $\gcd(a, N) = 1$. La condizione sulla coprimalità è essenziale: se infatti $\gcd(a, N) \neq 1$, avremmo già trovato un fattore non banale di N semplicemente calcolando il massimo comun divisore. Una volta scelto a, si considera la funzione esponenziale modulare definita come:

$$f(x) = a^x \bmod N$$

Questa funzione possiede una proprietà fondamentale: è periodica. Più precisamente, esiste un minimo intero $r > 0$ tale che:

$$a^r \equiv 1 \pmod{N}$$

Questo valore r è chiamato ordine di a modulo N. L'esistenza di tale periodo è garantita dal teorema di Eulero, secondo cui $a^{\phi(N)} \equiv 1 \pmod{N}$, dove $\phi(N)$ è la funzione di Eulero[NC10, Sezione A4.2]. Il periodo cercato è dunque un divisore di $\phi(N)$.

3.2.2 L'utilità del periodo per la fattorizzazione

A questo punto sorge naturale una domanda: perché la conoscenza del periodo r dovrebbe aiutarci a fattorizzare N? La risposta risiede in una proprietà algebrica profonda che collega l'ordine di un elemento alla struttura moltiplicativa del gruppo $(\mathbb{Z}/N\mathbb{Z})^*$[NC10, Sezione 5.3.2].

Se il periodo r soddisfa due condizioni specifiche, ovvero se r è pari e se $a^{r/2} \not\equiv -1$

(mod N), allora è possibile estrarre fattori non banali di N calcolando:

$$\gcd(a^{r/2} - 1, N) \quad \text{e} \quad \gcd(a^{r/2} + 1, N)$$

Questa è la chiave matematica dell'intero algoritmo[IBM24]. Per comprendere perché questa procedura funziona, è necessario esaminare più da vicino la struttura algebrica sottostante. Dalla relazione $a^r \equiv 1 \pmod{N}$ segue immediatamente che:

$$a^r - 1 \equiv 0 \pmod{N}$$

Poiché r è pari per ipotesi, possiamo fattorizzare il termine $a^r - 1$ utilizzando la differenza di quadrati[NC10, p. 233]:

$$a^r - 1 = (a^{r/2})^2 - 1 = (a^{r/2} - 1)(a^{r/2} + 1)$$

Ne consegue che:

$$(a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \pmod{N}$$

Questa congruenza ci dice che N divide il prodotto $(a^{r/2}-1)(a^{r/2}+1)$. Tuttavia, per ipotesi, $a^{r/2} \not\equiv -1 \pmod{N}$, il che significa che $a^{r/2} + 1$ non è divisibile per N. Analogamente, poiché r è il periodo minimo, anche $a^{r/2} - 1$ non può essere divisibile per N (altrimenti $r/2$ sarebbe il periodo). Di conseguenza, N divide il prodotto di due fattori senza dividere ciascun fattore singolarmente. Questo implica necessariamente che ciascuno dei due fattori condivide almeno un divisore primo con N, ma non tutti[NC10, Appendice 4.3, Teorema 5.2]. Il calcolo del massimo comun divisore permette quindi di estrarre questi divisori non banali.

3.2.3 Esempio numerico completo

Per rendere concreti i concetti esposti, consideriamo un esempio numerico completo. Supponiamo di voler fattorizzare $N = 15$. Scegliamo $a = 2$ e verifichiamo che $\gcd(2, 15) = 1$, come richiesto[IBM24]. Procediamo quindi al calcolo della sequenza di potenze di a modulo N:

x	$2^x \bmod 15$
1	2
2	4
3	8
4	1

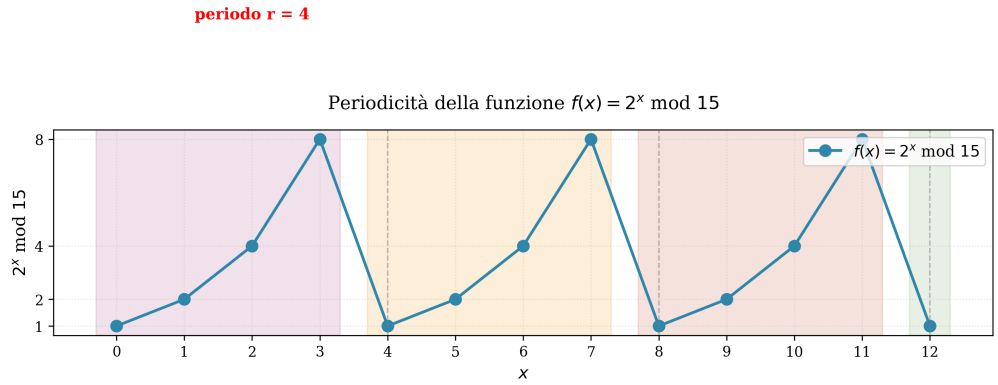


Figure 3.2: Visualizzazione della periodicità della funzione $f(x) = 2^x \bmod 15$. I colori evidenziano i cicli periodici di lunghezza $r = 4$. Elaborazione propria basata sull'esempio presentato da IBM Quantum Documentation.

Osserviamo che il valore si ripete dopo quattro iterazioni, pertanto il periodo è $r = 4$. Verifichiamo ora le condizioni necessarie: il periodo è effettivamente pari, e calcoliamo $a^{r/2} = 2^2 = 4$. Poiché $4 \not\equiv -1 \pmod{15}$ (infatti $4 \equiv 4 \pmod{15}$), entrambe le condizioni sono soddisfatte[NC10, Box 5.4]. Possiamo quindi applicare la formula per estrarre i fattori:

$$\gcd(4 - 1, 15) = \gcd(3, 15) = 3$$
$$\gcd(4 + 1, 15) = \gcd(5, 15) = 5$$

Abbiamo così ottenuto la fattorizzazione $15 = 3 \cdot 5$.

La complessità computazionale del metodo

A questo punto è importante sottolineare quali passaggi del metodo siano effettivamente computazionalmente trattabili e quale rappresenti invece il collo di bottiglia.

I passaggi che non presentano difficoltà computazionali significative includono il calcolo del massimo comun divisore, che può essere eseguito efficientemente tramite l'algoritmo di Euclide[NC10, Sezione A4.2], e l'esecuzione di esponenziazioni modulari, che può essere realizzata in tempo polinomiale mediante l'algoritmo di esponenziazione veloce[NC10, Box 5.2].

Il passaggio critico dell'intero procedimento è la determinazione del periodo r. Nel contesto della computazione classica, il calcolo del periodo della funzione $a^x \bmod N$ richiede essenzialmente di valutare la funzione ripetutamente fino a quando non si osserva la ripetizione del valore iniziale. Il numero di valutazioni necessarie nel caso peggiore può essere dell'ordine di N, e anche nel caso medio il numero di operazioni cresce in modo tale da rendere il problema intrattabile per valori di N sufficientemente grandi. È precisamente in questo passaggio che risiede il collo di bottiglia computazionale che impedisce alla fattorizzazione classica di essere efficiente.

Casi di fallimento e strategie di gestione

È importante osservare che il metodo descritto non ha sempre successo. Possono verificarsi due situazioni in cui la procedura non produce una fattorizzazione: il periodo r potrebbe risultare dispari, oppure potrebbe accadere che $a^{r/2} \equiv -1 \pmod{N}$. In entrambi questi casi, non è possibile applicare la fattorizzazione attraverso il massimo comun divisore nel modo descritto. Tuttavia, è possibile dimostrare che la probabilità di incorrere in uno di questi casi sfavorevoli è relativamente bassa[NC10, Teorema A4.13]. Inoltre, qualora si verifichi uno di questi casi, è sufficiente scegliere un valore diverso di a e ripetere l'intero procedimento[IBM24]. Con alta probabilità, dopo pochi tentativi si otterrà un periodo che soddisfa le condizioni necessarie.

Sintesi del contributo matematico

Ricapitolando, dal punto di vista strettamente matematico, l'algoritmo di Shor opera attraverso una sequenza logica di trasformazioni: in primo luogo, trasforma il problema della fattorizzazione in un problema di determinazione della periodicità di una funzione; in secondo luogo, sfrutta proprietà fondamentali dell'aritmetica modulare per collegare il periodo alla struttura dei fattori di N; in terzo luogo,

riduce il problema al calcolo dell'ordine di un elemento modulo N; infine, estrae i fattori cercati attraverso il calcolo del massimo comun divisore[NC10, Sezioni 5.3.1, 5.3.2, Teorema A4.11].

È fondamentale sottolineare che fino a questo punto non è stato introdotto alcun concetto di natura quantistica. Tutto quanto discusso appartiene al dominio della matematica classica e della teoria dei numeri. Il contributo essenziale della computazione quantistica risiede nella capacità di rendere efficiente il passaggio centrale, ovvero la determinazione del periodo[NC10, Sezione 5.3.1]. Mentre su un computer classico questo passaggio è computazionalmente intrattabile per numeri di grandi dimensioni, vedremo che un computer quantistico è in grado di eseguirlo in tempo polinomiale[NC10, p. 216], rendendo così l'intero algoritmo efficiente e ponendo una seria minaccia alla sicurezza dei sistemi crittografici basati sulla difficoltà della fattorizzazione.

3.2.4 La trasformata di Fourier quantistica

Per comprendere come un computer quantistico risolva efficientemente il problema della determinazione del periodo, è necessario introdurre uno strumento matematico fondamentale: la trasformata di Fourier[NC10, Sezione 5.1]. L'obiettivo di questa parte dell'analisi è chiarire perché la trasformata di Fourier rappresenti lo strumento naturale per affrontare problemi di periodicità e come la sua versione quantistica costituisca l'elemento centrale dell'algoritmo di Shor. Procederemo costruendo il ragionamento a partire da concetti elementari, senza assumere familiarità con gli aspetti tecnici della trasformata.

Il ruolo della trasformata di Fourier Siamo quindi arrivati al problema centrale: determinare il periodo della funzione $f(x) = a^x \bmod N$. Ricordiamo che il calcolo dei singoli valori di $f(x)$ non presenta difficoltà computazionali: dato un valore specifico di x, è possibile calcolare $f(x)$ in modo efficiente. La difficoltà risiede nella determinazione della frequenza con cui i valori della funzione si ripetono.

La trasformata di Fourier è uno strumento matematico che permette di operare un cambio di prospettiva fondamentale. Concettualmente, essa consente di passare da una descrizione di un segnale o di una funzione nel cosiddetto "dominio del

tempo" a una descrizione nel "dominio delle frequenze". Il periodo di una funzione è intrinsecamente una proprietà legata alle frequenze: una funzione periodica con periodo r può essere interpretata come un segnale che oscilla con frequenza fondamentale $1/r$. Questa osservazione costituisce la chiave per comprendere perché la trasformata di Fourier sia lo strumento appropriato per estrarre informazioni sulla periodicità.

Intuizione alla base della trasformata di Fourier Visto che l'ottica di questa ricerca è capire l'idea di fondo che si cela dietro shor, prima di vedere la formulazione matematica precisa, è utile sviluppare un'intuizione del funzionamento della trasformata di Fourier attraverso un esempio più semplice. Consideriamo un segnale periodico, che potrebbe rappresentare un'onda sonora, una vibrazione meccanica o qualsiasi altra grandezza fisica che varia nel tempo secondo un pattern ripetitivo. Nel dominio temporale, osserviamo come il segnale evolve istante per istante, registrando il valore della grandezza in funzione del tempo.

La trasformata di Fourier effettua un'operazione concettualmente diversa: prende il segnale nel dominio temporale e lo scompone in una somma di oscillazioni elementari, ciascuna caratterizzata da una frequenza ben definita. In altre parole, invece di descrivere quando il segnale assume determinati valori, la trasformata descrive quali frequenze compongono il segnale e con quale intensità ciascuna frequenza contribuisce. Il periodo del segnale originale emerge naturalmente da questa rappresentazione in termini di frequenze.

Per rendere questa idea più concreta, si consideri un segnale definito come la somma di due sinusoidi:

$$s(t) = \sin(2\pi t) + \sin(4\pi t)$$

Nel dominio temporale, questo segnale appare come un'onda dalla forma piuttosto irregolare, risultante dalla sovrapposizione delle due componenti. Tuttavia, nel dominio delle frequenze, la trasformata di Fourier rivela immediatamente la struttura sottostante: il segnale è composto da esattamente due componenti, una con frequenza 1 e una con frequenza 2. La trasformata di Fourier serve precisamente a estrarre questa decomposizione in componenti frequenziali, permettendo

di identificare le frequenze fondamentali che caratterizzano il segnale.

La trasformata di Fourier discreta Nel contesto dell'algoritmo di Shor, non abbiamo a che fare con segnali continui nel tempo, ma piuttosto con sequenze discrete di valori. La funzione $f(x) = a^x \bmod N$ produce una sequenza di valori interi $f(0), f(1), f(2), \ldots$ che si ripete con periodo r. Per analizzare questo tipo di segnali discreti è necessario utilizzare la trasformata di Fourier discreta, comunemente indicata con l'acronimo DFT (Discrete Fourier Transform)[NC10, Sezione 5.1].

La trasformata di Fourier discreta opera su una sequenza finita di valori $x_0, x_1, \ldots, x_{N-1}$ e la trasforma in un'altra sequenza $X_0, X_1, \ldots, X_{N-1}$, dove ciascun coefficiente X_k misura l'intensità con cui la frequenza k è presente nel segnale originale. La definizione formale della trasformata di Fourier discreta è data da:

$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-2\pi i k n / N}$$

Sebbene non sia necessario memorizzare questa formula nei dettagli, è importante comprenderne il significato concettuale. La sommatoria confronta il segnale originale con tutte le possibili oscillazioni discrete di frequenza k, e il valore di X_k misura quanto bene l'oscillazione di frequenza k si "accorda" con il segnale. Le oscillazioni che sono in fase con il segnale contribuiscono costruttivamente, mentre quelle fuori fase tendono a cancellarsi.

Periodicità e concentrazione spettrale Il collegamento diretto tra periodicità e trasformata di Fourier emerge quando si considera il caso di una funzione perfettamente periodica. Se una funzione possiede periodo r, la sua trasformata di Fourier non è distribuita uniformemente su tutte le frequenze, ma risulta concentrata su valori specifici. In particolare, l'energia spettrale è localizzata in corrispondenza di frequenze che sono multipli interi di $1/r$.

Questa proprietà implica che, osservando il risultato della trasformata di Fourier, è possibile ricavare informazioni precise sul periodo r. I picchi nella trasformata identificano le frequenze caratteristiche, e da queste frequenze si può risalire al periodo. Tuttavia, nel contesto della computazione classica, il calcolo della trasformata di Fourier discreta richiede un numero di operazioni che, anche utilizzando algoritmi

efficienti come la FFT (Fast Fourier Transform), cresce almeno proporzionalmente al numero di campioni. Per sequenze di lunghezza comparabile a N, questo rappresenta comunque un costo computazionale significativo che non risolve il problema della complessità della fattorizzazione.

3.2.5 Dalla DFT alla trasformata di Fourier quantistica

La trasformata di Fourier quantistica, comunemente indicata con l'acronimo QFT (Quantum Fourier Transform), rappresenta l'adattamento della trasformata di Fourier discreta al contesto della computazione quantistica[NC10, Sezione 5.2]. È fondamentale comprendere che la QFT non introduce nuove idee matematiche: si tratta essenzialmente della stessa trasformazione matematica definita dalla DFT, ma applicata a stati quantistici invece che a sequenze di numeri classici.

In un computer quantistico, una sequenza di valori viene rappresentata attraverso una sovrapposizione quantistica, ovvero uno stato della forma:

$$\sum_x \alpha_x |x\rangle$$

dove i coefficienti complessi α_x rappresentano le ampiezze di probabilità associate a ciascun valore di base $|x\rangle$. La QFT trasforma questo stato in un nuovo stato:

$$\sum_k \beta_k |k\rangle$$

dove i coefficienti β_k sono esattamente la trasformata di Fourier dei coefficienti originali α_x. In altre parole, la QFT realizza fisicamente, a livello di stato quantistico, la stessa operazione matematica che la DFT realizza a livello di array di numeri.

Il vantaggio computazionale della QFT La ragione per cui la QFT rappresenta un avanzamento rivoluzionario rispetto alla DFT classica risiede in una proprietà fondamentale della computazione quantistica: la capacità di operare simultaneamente su una sovrapposizione di stati[NC10, Sezioni 5.2.1, 5.2.2]. Nel computer quantistico, tutti i coefficienti α_x esistono contemporaneamente nella sovrapposizione, e la QFT agisce sull'intera sovrapposizione in un'unica operazione

coerente.

Il risultato di questa applicazione coerente della trasformata è che le ampiezze dei diversi stati interferiscono tra loro secondo le leggi della meccanica quantistica. Le frequenze che sono compatibili con la periodicità della funzione originale vengono amplificate attraverso **interferenza costruttiva**, mentre le frequenze incompatibili vengono soppresse attraverso **interferenza distruttiva**.

Quando si effettua una misura sullo stato risultante, si ottiene con alta probabilità un valore che contiene informazione sul periodo cercato.

Difatti questo è il principale snodo del teorema, il punto in cui interviene positivamente l'utilizzo dela quantistica, da qui in poi sarà necessario fare altre operazioni per estrarre correttamemnte l'informazione, infatti è importante sottolineare che la misura non fornisce direttamente il periodo r, ma piuttosto un valore dal quale è possibile estrarre r, ma il tutto semplicemente utilizzando tecniche matematiche classiche. In particolare, si utilizza l'algoritmo delle frazioni continue per approssimare il rapporto misurato con una frazione che ha il periodo come denominatore[NC10, Sezione 5.3.3]. Questa fase di post-elaborazione classica ci serve per completare l'algoritmo.

Esempio concettuale del meccanismo Per rendere più concreto il meccanismo di estrazione del periodo, consideriamo un esempio semplificato. Supponiamo che la funzione analizzata abbia periodo $r = 4$ e che il registro quantistico abbia dimensione Q. Dopo l'applicazione della QFT, lo stato quantistico non è distribuito uniformemente, ma presenta concentrazioni di ampiezza in corrispondenza di valori approssimabili come:

$$k \approx \frac{m}{4} \cdot Q$$

dove m è un intero. Quando si effettua la misura, si ottiene con alta probabilità uno di questi valori di k. Dal rapporto:

$$\frac{k}{Q} \approx \frac{m}{r}$$

è possibile ricostruire il periodo r attraverso l'approssimazione con frazioni continue[IBM24]. Questo esempio illustra il collegamento diretto tra l'applicazione della QFT, l'estrazione di informazione sulla periodicità attraverso la misura

quantistica, e il recupero finale del periodo che permette la fattorizzazione.

La natura del vantaggio quantistico Avendo capito il principio di funzionamento di shor possiamo chiarire meglio un aspetto concettuale che spesso genera confusione. Il computer quantistico non ottiene il suo vantaggio semplicemente "provando tutte le frequenze in parallelo" né "calcolando la trasformata esplicitamente come farebbe una CPU classica". Il meccanismo è qualitativamente diverso: il computer quantistico prepara uno stato quantistico con una struttura specifica, sfrutta le proprietà di interferenza e sovrapposizione per far emergere le frequenze rilevanti, e produce il periodo come risultato di un processo probabilistico governato dalle leggi della meccanica quantistica.

La QFT è il meccanismo matematico che rende possibile questa interferenza controllata. Essa permette di organizzare le ampiezze quantistiche in modo tale che, al momento della misura, i valori che contengono informazione sul periodo abbiano probabilità di osservazione significativamente maggiore rispetto agli altri. Questa capacità di manipolare coerentemente le ampiezze di probabilità attraverso interferenza quantistica è ciò che consente al computer quantistico di ottenere un vantaggio esponenziale rispetto ai metodi classici.

3.3 L'Algoritmo di Grover

A differenza dell'algoritmo di Shor, che si occupa di problemi specifici come la fattorizzazione e il logaritmo discreto, l'algoritmo di Grover affronta un problema più generale: la ricerca in un database non strutturato. Nonostante la sua generalità, l'algoritmo di Grover ha implicazioni significative per la crittografia, in particolare per i sistemi a chiave simmetrica, poiché fornisce un metodo più efficiente per eseguire attacchi brute-force. Cerchiamo quindi di capire come funziona l'algoritmo di Grover, partendo dai concetti matematici di base e arrivando a una comprensione intuitiva del suo meccanismo.

L'algoritmo di Grover, introdotto da Lov Grover nel 1996 [Gro96], rappresenta uno dei pilastri della computazione quantistica. A differenza dell'algoritmo di Shor, che offre un'accelerazione esponenziale rispetto alla controparte classica, Grover fornisce un'accelerazione quadratica nella risoluzione di problemi di ricerca non

strutturata.

Definizione del Problema

L'algoritmo affronta il problema di trovare un elemento specifico all'interno di un database non strutturato di dimensione $N = 2^n$. In un contesto classico, identificare l'unico elemento che soddisfa una determinata condizione richiederebbe mediamente $N/2$ operazioni (e N nel caso peggiore). L'algoritmo di Grover permette di ottenere il risultato in circa $\sqrt{N}$ passi, conseguendo così una riduzione quadratica della complessità computazionale.

Componenti Fondamentali

Il funzionamento dell'algoritmo si basa sulla manipolazione delle ampiezze di probabilità degli stati quantistici attraverso la ripetizione di un ciclo denominato *iterazione di Grover*. Gli elementi chiave sono i seguenti.

Inizializzazione. Il sistema viene preparato in una sovrapposizione uniforme di tutti i possibili N stati mediante l'applicazione della trasformata di Walsh-Hadamard $H^{\otimes n}$, garantendo che ogni stato $|x\rangle$ abbia la stessa ampiezza iniziale $1/\sqrt{N}$:

$$|\psi_0\rangle = H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle . \tag{3.1}$$

L'Oracolo $\mathcal{O}$. L'oracolo è una funzione unitaria in grado di riconoscere la soluzione del problema. Matematicamente, esso agisce invertendo la fase dell'ampiezza dello stato marcato $|\omega\rangle$, lasciando invariati tutti gli altri stati:

$$\mathcal{O} |x\rangle = (-1)^{f(x)} |x\rangle , \tag{3.2}$$

dove $f(x) = 1$ se $x = \omega$ (stato soluzione) e $f(x) = 0$ altrimenti.

L'Operatore di Diffusione $\mathcal{D}$. L'operatore di diffusione, definito come

$$\mathcal{D} = 2 |\psi_0\rangle \langle\psi_0| - I, \tag{3.3}$$

esegue un'operazione nota come *inversione rispetto alla media* (inversion about average). Il suo effetto è quello di amplificare l'ampiezza dello stato marcato dall'oracolo e di attenuare quella degli stati rimanenti, sfruttando i fenomeni di interferenza costruttiva e distruttiva.

Esempio Numerico: Ricerca su 2 Qubit ($N = 4$)

Consideriamo il caso ideale in cui il database ha $N = 4$ elementi ($n = 2$ qubit) e una sola soluzione ($M = 1$). In questo scenario, il numero ottimale di iterazioni di Grover è $k = \lfloor \frac{\pi}{4}\sqrt{4} \rfloor = 1$, dunque è sufficiente una singola iterazione per ottenere la soluzione con probabilità massima.

1. Inizializzazione. Applichiamo la trasformata di Walsh-Hadamard $H^{\otimes 2}$ allo stato $|00\rangle$, ottenendo la sovrapposizione uniforme:

$$|\psi_0\rangle = \frac{1}{2}\big(|00\rangle + |01\rangle + |10\rangle + |11\rangle\big). \tag{3.4}$$

Tutte le ampiezze iniziali sono uguali: $\alpha_i = \frac{1}{2} = 0.5$.

2. Applicazione dell'Oracolo. Supponiamo che la soluzione sia lo stato $|11\rangle$. L'oracolo $\mathcal{O}$ inverte il segno dell'ampiezza di tale stato, lasciando invariati gli altri:

$$|\psi_1\rangle = \mathcal{O}\,|\psi_0\rangle = \frac{1}{2}\,|00\rangle + \frac{1}{2}\,|01\rangle + \frac{1}{2}\,|10\rangle - \frac{1}{2}\,|11\rangle\,. \tag{3.5}$$

Il vettore delle ampiezze è ora $\{\,0.5,\ 0.5,\ 0.5,\ -0.5\,\}$.

3. Inversione rispetto alla Media. Applichiamo l'operatore di diffusione $\mathcal{D}$, che realizza l'inversione rispetto alla media delle ampiezze.

(i) Calcoliamo la media delle ampiezze correnti:

$$\langle\alpha\rangle = \frac{1}{4}\big(0.5 + 0.5 + 0.5 + (-0.5)\big) = \frac{1}{4}\cdot 1.0 = 0.25. \tag{3.6}$$

(ii) Applichiamo la trasformazione $\alpha_i \mapsto 2\langle\alpha\rangle - \alpha_i$ a ciascuna ampiezza:

- Stati non soluzione ($\alpha_i = 0.5$): $2(0.25) - 0.5 = 0.0$.
- Stato soluzione ($\alpha_i = -0.5$): $2(0.25) - (-0.5) = 1.0$.

4. Risultato. Lo stato del sistema dopo una singola iterazione di Grover è:

$$|\psi_2\rangle = \mathcal{D}\,|\psi_1\rangle = |11\rangle\,. \tag{3.7}$$

L'ampiezza dello stato soluzione è stata amplificata a 1.0, mentre tutte le ampiezze degli stati non soluzione sono state annullate. La probabilità di misurare la soluzione corretta è pertanto del 100% con una singola interrogazione all'oracolo. A titolo di confronto, un algoritmo classico richiederebbe in media $\frac{N+1}{2} = 2.5$ tentativi per individuare la soluzione in un database di 4 elementi.

Interpretazione Geometrica

L'algoritmo di Grover ammette un'elegante interpretazione geometrica. Lo spazio di Hilbert rilevante può essere ridotto a un sottospazio bidimensionale, definito dal vettore soluzione $|\omega\rangle$ e dal vettore

$$|s'\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq \omega} |x\rangle\,, \tag{3.8}$$

che rappresenta la sovrapposizione uniforme di tutti gli stati non risolutivi. In questo spazio, ogni iterazione di Grover $G = \mathcal{D}\mathcal{O}$ corrisponde a una rotazione del vettore di stato di un angolo fisso

$$\theta = 2\arcsin\left(\sqrt{\frac{M}{N}}\right), \tag{3.9}$$

dove M è il numero di soluzioni valide. Dopo circa

$$k \approx \frac{\pi}{4}\sqrt{\frac{N}{M}} \tag{3.10}$$

iterazioni, il vettore di stato risulta quasi perfettamente allineato con $|\omega\rangle$, e una misurazione nel base computazionale restituisce la soluzione con alta probabilità.

Rilevanza per la Crittografia

In ambito crittografico, l'algoritmo di Grover può essere impiegato per condurre attacchi di tipo *brute force* contro cifrari a chiave simmetrica. Un avversario quantistico in grado di eseguire l'algoritmo di Grover riduce infatti la complessità di una ricerca esaustiva sullo spazio delle chiavi da $\mathcal{O}(2^n)$ a $\mathcal{O}(2^{n/2})$. A titolo esemplificativo, mentre un computer classico dovrebbe testare fino a 2^{128} combinazioni per violare AES-128, l'algoritmo di Grover riduce lo sforzo computazionale a $\mathcal{O}(2^{64})$ operazioni quantistiche.

Per tale ragione, nell'ottica della transizione verso la crittografia post-quantistica, è ritenuto necessario raddoppiare la lunghezza delle chiavi simmetriche — ad esempio, adottando AES-256 in luogo di AES-128 — al fine di preservare un livello di sicurezza equivalente contro un avversario dotato di capacità quantistiche.

3.4 Considerazioni finali

Riassumendo brevemente,

- abbiamo introdotto i concetti di meccanica quantistica necessari a comprendere il funzionamento di un computer quantistico;
- abbiamo convertito il problema della fattorizzazione in un problema di determinazione del periodo di una funzione esponenziale modulare;
- abbiamo poi trovato un algoritmo per calcolarne il perido, e l'abbiamo ottimizzato sostituendo la DFT (che costituiva il contributo con complessità computazionalemnte non polinomiale) con la QFT per ottenere un vantaggio esponenziale rispetto ai metodi classici, permettendo così di risolvere il problema della fattorizzazione in tempo polinomiale su un computer quantistico.
- abbiamo infine introdotto l'algoritmo di Grover, che fornisce un'accelerazione quadratica per problemi di ricerca non strutturata, con implicazioni dirette sulla sicurezza dei sistemi a chiave simmetrica.

Ora che conosciamo il funzionamento matematico/teorico gli algoritmi di Shor e Grover, possiamo passare alla prossima fase della trattazione: analizzare la risposta

della comunità crittografica a questa minaccia: la crittografia post-quantistica.

Chapter 4

Crittografia post quantistica

La crittografia post-quantistica, anche nota come crittografia quantum-safe o quantum-resistant, è quell'ambito della ricerca e dello sviluppo di algoritmi crittografici (solitamente a chiave pubblica) progettati per essere sicuri contro attacchi crittanalitici effettuati da computer quantistici. Ecco i pilastri fondamentali che compongono questa definizione:

1. Indipendenza dall'hardware quantistico. A differenza della "crittografia quantistica"[1] (che sfrutta i principi della fisica quantistica per proteggere le comunicazioni, come la distribuzione quantistica delle chiavi), la crittografia post-quantistica si basa su problemi matematici eseguiti su computer classici, ma strutturati in modo da essere resistenti anche ai futuri computer quantistici.

2. Superamento della vulnerabilità agli algoritmi quantistici

   La crittografia post-quantistica deve essere immune alle due minacce principali rappresentate dai computer quantistici:

   - Algoritmo di Shor: di cui abbiamo ampiamente discusso nel capitolo precedente.
   - Algoritmo di Grover: in grado di fornire un'accelerazione quadratica per le ricerche brute-force non strutturate. Per contrastare Grover, la PQC

[1] Quantum Computing and Cryptography

simmetrica richiede semplicemente il raddoppio della lunghezza delle chiavi (ad esempio, passare da AES-128 a AES-256) per mantenere lo stesso livello di sicurezza.

3. Fondazione su nuovi problemi matematici: La PQC si basa su problemi matematici che, allo stato attuale della ricerca, non presentano vulnerabilità esponenziali quantistiche. Nel prossimo capitolo: "Algoritmi post-quantistici" esamineremo nel dettaglio le principali famiglie di problemi matematici su cui si basano gli algoritmi post-quantistici.

4. "Forward Secrecy" Una definizione completa di PQC include la necessità di proteggere i dati non solo in futuro, ma anche oggi.

   Il modello di minaccia "Harvest Now, Decrypt Later" (raccogli ora, decifra dopo) suggerisce che attori malintenzionati possano archiviare dati criptati oggi per decifrarli quando saranno disponibili computer quantistici sufficientemente potenti.

   Per questo motivo è considerata una priorità di sicurezza nazionale e infrastrutturale immediatarisulta e non così lontana. Queste motivazioni infatti stanno guidando la comunità scentifica a standardizzare quanto prima possibile questi algoritmi in modo da iniziare una migrazione verso questi protocolli già oggi.

4.1 Requisiti e obiettivi

Requisiti pratici e tempistiche Sebbene la minaccia sia teoricamente dimostrata, la realizzazione pratica di computer quantistici capaci di violare RSA ed ECC richiede risorse considerevoli. Secondo stime del NIST, per compromettere una chiave RSA-2048 sarebbero necessari diversi milioni di qubit logici affidabili, mentre le implementazioni attuali (2024) operano con centinaia di qubit fisici caratterizzati da elevati tassi di errore.

La transizione da qubit fisici a qubit logici richiede tecniche di correzione degli errori quantistici che impongono un overhead significativo: potrebbero essere

necessari da centinaia a migliaia di qubit fisici per realizzare un singolo qubit logico stabile.

4.2 Ruolo del Nist e processo di standardizzazione

La risposta: crittografia post-quantistica Di fronte a questa minaccia emergente, il National Institute of Standards and Technology (NIST) ha avviato nel 2016 un processo di standardizzazione per identificare algoritmi crittografici resistenti agli attacchi quantistici. Nel luglio 2022, il NIST ha annunciato i primi algoritmi selezionati per la standardizzazione, basati su problemi matematici ritenuti difficili anche per computer quantistici, come i reticoli algebrici e i codici correttori di errori.

La migrazione verso la crittografia post-quantistica rappresenta una delle sfide più urgenti per la sicurezza informatica moderna, richiedendo un'attenta pianificazione per sostituire l'infrastruttura crittografica esistente mantenendo retrocompatibilità e garantendo una transizione graduale e sicura.

Chapter 5

Algoritmi post-quantistici

A questo punto della relazione, abbiamo compreso come funziona la crittografia classica e come i computer quantisci sono in grado di violarla, ora possiamo quindi entrare nel secondo punto fondamentale di questo elaborato: le famiglie di algoritmi (PQC). Prima però servirà un ultimo preambolo, visto che abbiamo compreso che l'interra crittografia si basa su problemi matematici, analizziamo in breve quali sono le classi di complessità computazionali dei nuovi problemi matematici che andremo poi ad analizzare per capirne la sicurezza.

Introduzione

Classi di complessità e problemi matematici Per comprendere perché la crittografia post-quantistica rappresenta una soluzione efficace, introduciamo la gerarchia delle classi di complessità computazionale per capire dove si collocano i diversi problemi che ci saranno utili in seguito.

Il problema P vs NP Uno dei problemi ancora irrisolti nella matematica e informatica teorica è appunto il problema P vs NP, difatti è inserito tra i sette problemi del millennio dal Clay Mathematics Institute[1]. La questione riguarda la differenza tra il "risolvere" un problema e il "verificare" una soluzione già data.

- **P (Polynomial time)**: Problemi che possono essere risolti in tempo polinomiale da un algoritmo deterministico. Questi problemi sono considerati

[1] Clay Mathematics Institute

"facili" da risolvere. Esempio: ordinamento di una lista.

- **NP (Nondeterministic Polynomial time)**: Problemi per i quali, data una possibile soluzione, è possibile verificare la correttezza in tempo polinomiale. Esempio: il problema del cammino hamiltoniano.

- **NP-hard**: Problemi almeno tanto difficili quanto i problemi più difficili in NP. Formalmente, un problema è NP-hard se ogni problema in NP può essere ridotto a esso in tempo polinomiale; pertanto, trovare un algoritmo polinomiale per un problema NP-hard implicherebbe che $P = NP$. Esempio: il problema del commesso viaggiatore (TSP).

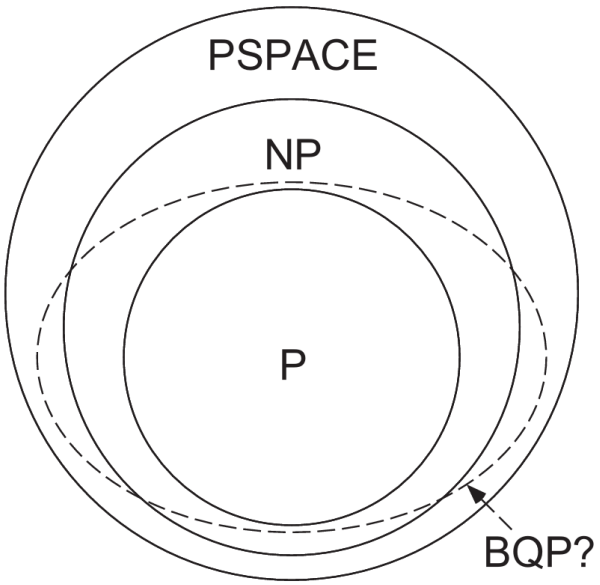


Figure 5.1: Relazione tra le classi di complessità classiche (P, NP, PSPACE) e la classe quantistica BQP.

Da qui emerge una domanda cruciale: per ogni problema la cui soluzione è facile da verificare (NP) è anche facile trovare una soluzione (P)? Se $P = NP$, allora ogni volta che possiamo controllare rapidamente una soluzione deve esistere anche un modo veloce per trovarla. La comunità scientifica concorda che $P \neq NP$, il che implica che esistono problemi intrinsecamente difficili per i quali trovare la

soluzione richiede tempi esponenziali, anche se verificarli è immediato, e questo rappresenta il fondamento della sicurezza crittografica.

BQP Nel caso della nostra analisi, è importante introdurre in questa gerarchia anche la classe BQP (Bounded-error Quantum Polynomial time), che rappresenta l'insieme dei problemi risolvibili efficientemente da un computer quantistico. Attualmente, sebbene non sia dimostrato, la comunità scientifica ritiene che BQP non contenga i problemi NP-hard. In altre parole, si ipotizza che nemmeno un computer quantistico possa risolvere in modo efficiente problemi come il TSP. Questo è fondamentale perché implica che esistono problemi matematici che rimangono difficili da risolvere anche rispetto al calcolo quantistico e che quindi possono essere utilizzati come base per la crittografia post-quantistica. In questa classe di problemi (BQP) rientrano infatti RSA e ECC che, come abbiamo visto, sono vulnerabili per via della loro struttura matematica basata sulla periodicità.

PQC La crittografia post-quantistica sposta quindi la sua sicurezza dai problemi di classe NP-Intermediate (di cui si ritiene facciano parte RSA ed ECC) a nuove famiglie matematiche che, allo stato attuale della ricerca, non presentano vulnerabilità esponenziali quantistiche; in particolare, molti di questi schemi sono legati a varianti di problemi NP-hard.

5.1 Lattice-baces

La crittografia basta sui reticoli (Lattice-based)[2] è una delle famiglie più promettenti della crittografia post-quantistica. La sua sicurezza si basa sulle proprietà geometriche dei reticoli. Un reticolo è un insieme di punti nello spazio n-dimensionale che possono essere rappresentati come combinazioni lineari di vettori base con coeficenti interi.

Il Problema matematico fondamentale La sicurezza di questa famiglia di algoritmi si basa sulla difficoltà di risolvere problemi specifici all'interno di questi reticoli. Questo è il campo da gioco

[2]NIST FIPS 203: Lattice-Based Cryptography

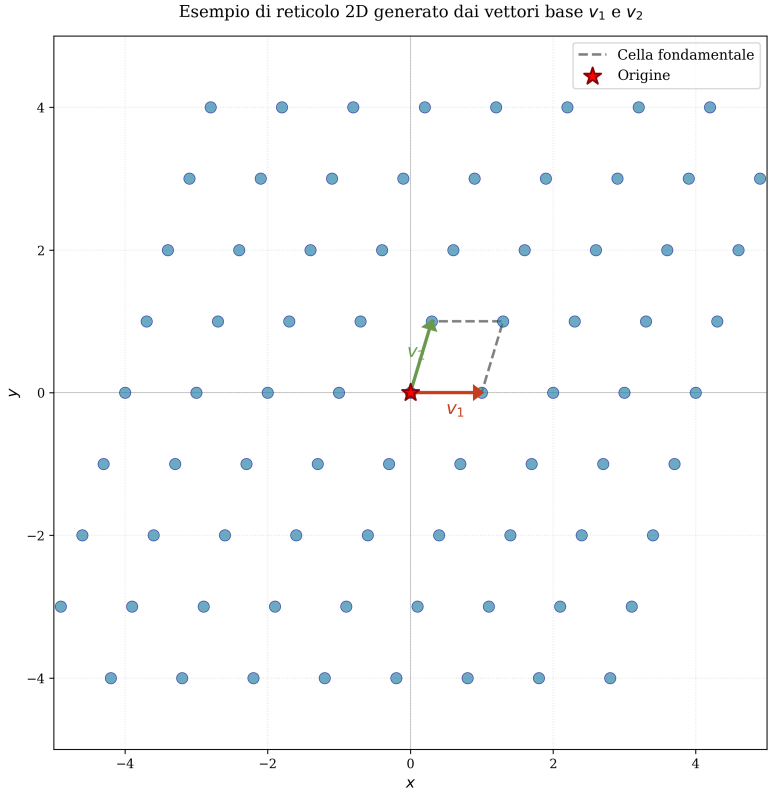


Figure 5.2: Rappresentazione bidimensionale di un reticolo. I punti verdi rappresentano i vettori del reticolo, mentre i vettori blu rappresentano la base del reticolo.

I principali problemi che ne derivaono sono:

5.1.1 Learning With Errors (LWE)

Introdotto da Oded Regev nel 2005, per il quale ha ricevuto il premio Gödel nel 2018, LWE[3] consiste nel risalire ad un vettore segreto $\mathbf{s} \in \mathbb{Z}_q^n$ dato un insieme di equazioni lineari rumorose.

Formalmente:

- si sceglie una matrice pubblica $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ con elementi scelti uniformemente a caso
- ognuno sceglie un vettore segreto $\mathbf{s} \in \mathbb{Z}_q^n$ (la chiave privata)

[3] LWE

- e un vettore di errore $\mathbf{e} \in \mathbb{Z}_q^m$ con componenti piccole

Il problema fornisce coppie $(\mathbf{A}, \mathbf{b})$ dove:

$$\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \pmod{q} \tag{5.1}$$

L'obiettivo, come abbiamo anticipato, è recuperare il vettore segreto conoscendo solo $\mathbf{A}$ e $\mathbf{b}$. Non conoscere l'errore $\mathbf{e}$ rende il problema computazionalmente intrattabile. La difficoltà del problema è stata dimostrato che è correlata alla risoluzione di problemi nel caso pessimo sui reticoli, come come il Shortest Vector Problem (SVP) e il Shortest Independent Vectors Problem (SIVP) classificati come NP-hard. Esisitono due versioni del problema:

- **Search-LWE**: Trovare il vettore segreto $\mathbf{s}$ dato un insieme di campioni
- **Decision-LWE**: Distinguere campioni LWE $(A, \mathbf{As} + \mathbf{e})$ da campioni completamente casuali $(A, \mathbf{u})$ dove $\mathbf{u}$ è uniforme

È stato dimostrato che le due versioni sono equivalenti, risolvere il problema decisionale consente di risolvere anche quello di ricerca.

5.1.2 Module Learning with Errors (MLWE)

È una generalizzazione di LWE che opera su strutture chiamate "moduli" su anelli polinomiali, tipicamente $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ dove n è una potenza di 2. Questa variante è utilizzata negli standard moderni perché permette una maggiore efficienza computazionale e chiavi di dimensioni significativamente ridotte rispetto al LWE standard.

La formulazione MLWE sostituisce i vettori con vettori di polinomi e le matricrici con matrici di polinomi, mantenendo la stessa struttura generale ma sfruttando la struttura algebrica degli anelli per migliorare le prestazioni. L'equazione diventa:

$$\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \pmod{q} \tag{5.2}$$

dove ora $\mathbf{A}$, $\mathbf{s}$ e $\mathbf{e}$ sono elementi del modulo su R_q.

5.1.3 Short Integer Solution (SIS/MSIS)

Consiste nel trovare una soluzione "piccola" (con coefficienti bassi) per un sistema lineare. Data una matrice $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, trovare un vettore non nullo $\mathbf{x}$ con norma piccola tale che $\mathbf{Ax} = \mathbf{0} \pmod{q}$. Questo problema è utilizzato principalmente per schemi di firma digitale.

Resistenza agli attacchi quantistici

I problemi basati sui reticoli (come MLWE e SIS) sono ritenuti difficili da risolvere anche per un avversario dotato di un computer quantistico a tolleranza d'errore. Al momento non sono noti algoritmi quantistici in grado di rompere efficientemente questi schemi. L'algoritmo quantistico di Grover può fornire solo un'accelerazione quadratica nella ricerca, che è significativa ma non sufficiente a rendere il problema trattabile.

La resistenza quantistica deriva dalla natura "disordinata" e non strutturata dei problemi sui reticoli, in contrasto con la struttura periodica che caratterizza i problemi di fattorizzazione e logaritmo discreto.

5.1.4 Key-Encapsulation Mechanism (KEM)

Un Key-Encapsulation Mechanism è un insieme di algoritmi che, sotto determinate condizioni, può essere utilizzato da due parti per stabilire una chiave segreta condivisa su un canale pubblico[4]. A differenza della crittografia a chiave pubblica tradizionale come RSA, dove un messaggio può essere cifrato direttamente con la chiave pubblica, un KEM è progettato specificamente per lo scambio sicuro di chiavi simmetriche.

Il protocollo KEM basato su MLWE funziona secondo il seguente schema:

Generazione delle chiavi (KeyGen) Alice genera una coppia di chiavi:

1. Sceglie un vettore segreto $\mathbf{s}$ (piccolo, campionato da una distribuzione di errore)

[4]NIST FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard

2. Genera una matrice pubblica $\mathbf{A}$ (uniforme casuale)

3. Campiona un vettore di errore $\mathbf{e}$ (piccolo)

4. Calcola la chiave pubblica: $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$

La **chiave di incapsulamento** (pubblica) è la coppia $(\mathbf{A}, \mathbf{t})$, mentre la **chiave di decapsulamento** (privata) è $\mathbf{s}$.

Incapsulamento (Encaps) Bob, ricevuta la chiave pubblica di Alice, vuole stabilire una chiave condivisa:

1. Campiona un nuovo vettore segreto temporaneo $\mathbf{r}$ e vettori di errore $\mathbf{e}_1$, $\mathbf{e}_2$

2. Calcola il ciphertext:

$$\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 \pmod{q} \tag{5.3}$$
$$v = \mathbf{t}^T\mathbf{r} + \mathbf{e}_2 + \text{Encode}(K) \pmod{q} \tag{5.4}$$

   dove K è la chiave segreta condivisa (tipicamente 256 bit) codificata opportunamente

3. Invia il ciphertext $(\mathbf{u}, v)$ ad Alice

Decapsulamento (Decaps) Alice, ricevuto il ciphertext da Bob, recupera la chiave condivisa:

1. Calcola: $w = v - \mathbf{s}^T\mathbf{u} \pmod{q}$

2. Decodifica w per ottenere K

Correttezza del protocollo La correttezza si basa sul fatto che:

$$w = v - \mathbf{s}^T\mathbf{u} \tag{5.5}$$
$$= (\mathbf{t}^T\mathbf{r} + \mathbf{e}_2 + \text{Encode}(K)) - \mathbf{s}^T(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1) \tag{5.6}$$
$$= ((\mathbf{A}\mathbf{s} + \mathbf{e})^T\mathbf{r} + \mathbf{e}_2 + \text{Encode}(K)) - \mathbf{s}^T\mathbf{A}^T\mathbf{r} - \mathbf{s}^T\mathbf{e}_1 \tag{5.7}$$
$$= \mathbf{e}^T\mathbf{r} + \mathbf{e}_2 - \mathbf{s}^T\mathbf{e}_1 + \text{Encode}(K) \tag{5.8}$$

Il termine di errore totale $\mathbf{e}^T\mathbf{r}+\mathbf{e}_2-\mathbf{s}^T\mathbf{e}_1$ rimane sufficientemente piccolo (perché tutti i vettori coinvolti hanno componenti piccole) da permettere la decodifica corretta di K.

Esempio di Numerico semplificato

Per chiarire meglio come ML-KEM riesca a trasmettere dati in modo sicuro nonostante la presenza di "rumore" matematico proviamo ad analizzare in concreto l'algoritmo con un esempio numerico semplificato attraversando le tre fasi: generazione delle chiavi, incapsulamento e decapsulamento..

Specifico che in ML-KEM, tutte le operazioni avvengono all'interno di un anello di polinomi con coefficienti nel campo $\mathbb{Z}_q$ dove $q = 3329$. Per semplicità di calcolo, consideriamo un sistema con parametri semplificati:

- Modulo: $q = 17$
- Segreto di Alice (s): 3
- Errore di Alice (e): 1
- Matrice pubblica (A): 7 (nel mondo reale è una matrice, qui un singolo valore)

Fase 1: Generazione delle Chiavi (Alice) Alice prepara la sua chiave pubblica secondo il protocollo ML-KEM:

1. Calcola $t = As + e \pmod q$:

$$t = (7 \times 3) + 1 = 21 + 1 = 22$$
$$t \equiv 5 \pmod{17}$$

2. La **chiave pubblica** è $(A = 7, t = 5)$
3. La **chiave privata** è $s = 3$

Fase 2: Incapsulamento (Bob) Bob vuole stabilire una chiave condivisa con Alice inviando il bit $K = 1$.

1. Codifica il messaggio: Encode(1) $= \lfloor q/2 \rfloor = 8$
2. Sceglie parametri casuali:
   - Numero segreto temporaneo: $r = 2$
   - Errori: $e_1 = 1$, $e_2 = 1$
3. Calcola il ciphertext (u, v):

$$\begin{aligned} u &= Ar + e_1 = (7 \times 2) + 1 = 14 + 1 = 15 \\ v &= tr + e_2 + \text{Encode}(K) \\ &= (5 \times 2) + 1 + 8 = 10 + 1 + 8 = 19 \\ &\equiv 2 \pmod{17} \end{aligned}$$

4. Invia ad Alice il ciphertext: $(u = 15, v = 2)$

Fase 3: Decapsulamento (Alice) Alice riceve $(u = 15, v = 2)$ e usa la sua chiave privata $s = 3$ per recuperare il bit originale.

1. Calcola $w = v - su \pmod{q}$:

$$\begin{aligned} w &= 2 - (3 \times 15) \pmod{17} \\ &= 2 - 45 \pmod{17} \\ &= -43 \pmod{17} \\ &= 8 \pmod{17} \end{aligned}$$

2. Decodifica w: Alice applica la funzione di decodifica confrontando $w = 8$ con i valori attesi 0 e $\lfloor q/2 \rfloor = 8$. Poiché w corrisponde esattamente a Encode(1) $= 8$, la decodifica restituisce $K = 1$.

La chiave condivisa $K = 1$ è stata trasmessa con successo!

Analisi della Correttezza Per comprendere perché il protocollo funziona, espandiamo il calcolo di Alice sostituendo $t = As + e$:

$$\begin{aligned} w &= v - su \\ &= (tr + e_2 + \text{Encode}(K)) - s(Ar + e_1) \\ &= (As + e)r + e_2 + \text{Encode}(K) - sAr - se_1 \\ &= \text{Encode}(K) + \underbrace{(er + e_2 - se_1)}_{\text{Rumore Totale}} \end{aligned}$$

Verifichiamo il rumore totale nel nostro esempio:

$$\begin{aligned} \text{Rumore} &= er + e_2 - se_1 \\ &= (1 \times 2) + 1 - (3 \times 1) \\ &= 2 + 1 - 3 = 0 \end{aligned}$$

Nel nostro esempio il rumore si è annullato perfettamente, producendo $w = 8 + 0 = 8$. Nella realtà, i parametri e, r, e_1, e_2 sono scelti da distribuzioni binomiali centrate che garantiscono che il rumore totale rimanga sempre al di sotto di una soglia critica (tipicamente $< q/4$), permettendo ad Alice di decodificare correttamente il bit anche in presenza di piccole perturbazioni.

Gestione del Rumore e Affidabilità Se il rumore fosse stato diverso da zero, ad esempio +2, Alice avrebbe ottenuto $w = 10$ invece di 8. Applicando la funzione di decodifica:

$$\text{Bit} = \left\lfloor \frac{2 \cdot 10}{17} \right\rceil \pmod 2 = \lfloor 1.17 \rceil = 1$$

Il sistema sarebbe comunque riuscito a recuperare correttamente il bit grazie al margine di tolleranza integrato nel design. Solo se il rumore superasse la soglia di $q/4 \approx 4.25$ si verificherebbe un errore di decodifica (*decapsulation failure*).

Parametri Reali e Considerazioni Implementative Nei sistemi ML-KEM reali (FIPS 203):

- Il modulo è $q = 3329$
- A è una matrice di polinomi di grado 255
- s, e, r, e_1, e_2 sono vettori di polinomi campionati da distribuzioni binomiali centrate
- Il valore per il bit 1 è Encode(1) = 1665
- Il rumore è progettato per rimanere molto al di sotto della soglia critica $q/4 \approx 832$
- La probabilità di *decapsulation failure* per ML-KEM-768 è circa $2^{-164.8}$
- Per velocizzare le moltiplicazioni polinomiali (come $s^T u$) si utilizza la Number-Theoretic Transform (NTT), riducendo la complessità da $O(n^2)$ a $O(n \log n)$
- Lo schema utilizza la trasformata di Fujisaki-Okamoto per raggiungere la sicurezza IND-CCA2

Sicurezza Un attaccante che intercetta $(\mathbf{A}, \mathbf{t})$ e $(\mathbf{u}, v)$ deve quindi riuscire a risolvere il problema MLWE per recuperare K, che è computazionalmente intrattabile. La chiave segreta condivisa può poi essere utilizzata con algoritmi crittografici simmetrici (come AES) per cifrare e autenticare le comunicazioni.

Firme Digitali basate su reticoli

Le firme digitali sono utilizzate per autenticare l'identità e l'integrità dei dati. Inoltre, il destinatario di dati firmati può utilizzare una firma digitale come prova per dimostrare a terzi che la firma è stata effettivamente generata dal firmatario dichiarato (proprietà di non ripudio)[5].

Gli schemi di firma basati su reticoli utilizzano tipicamente il problema SIS/MSIS come fondamento della loro sicurezza.

[5]NIST FIPS 204: Module-Lattice-Based Digital Signature Standard

Stato della standardizzazione

Il NIST (National Institute of Standards and Technology) ha avviato un processo di standardizzazione della crittografia post-quantistica nel 2016[6]. Il 13 agosto 2024 sono stati pubblicati i primi standard finali basati sui reticoli[7]:

- **FIPS 203 (ML-KEM)**: Basato sull'algoritmo CRYSTALS-Kyber, è lo standard primario per lo scambio di chiavi. ML-KEM è l'acronimo di Module-Lattice-Based Key-Encapsulation Mechanism. La sicurezza di ML-KEM è correlata alla difficoltà computazionale del problema Module Learning with Errors. Attualmente si ritiene che ML-KEM sia sicuro anche contro avversari in possesso di un computer quantistico.
- **FIPS 204 (ML-DSA)**: Basato su CRYSTALS-Dilithium, è lo standard primario per le firme digitali. ML-DSA è l'acronimo di Module-Lattice-Based Digital Signature Algorithm. Si ritiene che ML-DSA sia sicuro anche contro avversari in possesso di un computer quantistico su larga scala.
- **FIPS 206 (FN-DSA)**: Basato sull'algoritmo FALCON, è un ulteriore standard per firme digitali attualmente in fase di standardizzazione finale. FN-DSA è l'acronimo di FFT (Fast-Fourier Transform) over NTRU-Lattice-Based Digital Signature Algorithm[8]. FALCON utilizza reticoli NTRU e, a differenza degli altri algoritmi selezionati, si basa sull'aritmetica in virgola mobile. Offre firme molto compatte e prestazioni elevate, rendendolo particolarmente adatto a scenari in cui la larghezza di banda è limitata o la velocità è critica.

Gli standard possono e devono essere messi in uso ora. Le organizzazioni sono incoraggiate a iniziare la migrazione verso questi sistemi per proteggersi dalla futura minaccia quantistica.

Parametri e livelli di sicurezza

All'interno degli standard esistono diversi set di parametri che offrono compromessi tra sicurezza e prestazioni:

[6]NIST: Post-Quantum Cryptography Standardization
[7]NIST: NIST Releases First 3 Finalized Post-Quantum Encryption Standards
[8]NIST: FIPS 206 FN-DSA (FALCON)

ML-KEM (ex CRYSTALS-Kyber) Questo standard specifica tre set di parametri per ML-KEM. In ordine di crescente forza di sicurezza e decrescente prestazione, questi sono ML-KEM-512, ML-KEM-768 e ML-KEM-1024:

- **ML-KEM-512**: Livello di sicurezza Categoria 1 (equivalente a AES-128), chiave di incapsulamento di 800 bytes, chiave di decapsulamento di 1632 bytes, ciphertext di 768 bytes.

- **ML-KEM-768**: Livello di sicurezza Categoria 3 (equivalente a AES-192), raccomandato come default dal NIST. Chiave di incapsulamento di 1184 bytes, chiave di decapsulamento di 2400 bytes, ciphertext di 1088 bytes. Offre un ottimo equilibrio tra sicurezza e velocità.

- **ML-KEM-1024**: Livello di sicurezza Categoria 5 (equivalente a AES-256), massima sicurezza. Chiave di incapsulamento di 1568 bytes, chiave di decapsulamento di 3168 bytes, ciphertext di 1568 bytes. Prestazioni ridotte e chiavi più grandi.

Tutti e tre i set di parametri producono una chiave segreta condivisa di 32 bytes (256 bit).

ML-DSA (ex CRYSTALS-Dilithium) Esistono tre versioni: ML-DSA-44, ML-DSA-65 e ML-DSA-87, dove i numeri si riferiscono alle dimensioni della matrice utilizzata nell'algoritmo (rispettivamente matrici 4×4, 6×5 e 8×7), corrispondenti a diversi livelli di sicurezza (rispettivamente Categorie 2, 3 e 5).

Algoritmi alternativi Esistono anche altri algoritmi basati sui reticoli che non sono stati selezionati come standard primari ma che sono serviti come candidati o alternative, come NTRU, SABER e FrodoKEM. Ad esempio, NTRU è basato su problemi di reticoli ma con una struttura matematica differente che lo rende una potenziale alternativa in caso di vulnerabilità scoperte in ML-KEM.

5.2 Hash-based Cryptography

Fin'ora abbiamo parlato principalmente di crittografia a chiave pubblica per la riservatezza e cifratura, ma la crittografia ha anche un ruolo fondamentale nell'**autenticità** e **integrità** dei dati.

La firma digitale è il concetto che risolve entrambi i problemi. L'idea di fondo è elegante: si usa una funzione matematica che lega in modo indissolubile il contenuto del documento alla chiave privata del mittente, producendo qualcosa che chiunque può verificare con la corrispondente chiave pubblica, ma che solo il mittente può produrre. Nello scenario classico questo concetto era interconnesso con la crittografia a chiave pubblica, ma con l'avvento della minaccia quantistica è diventato necessario ripensare completamente questo paradigma.

Per comprendere i nuovi standard in questo ambito, è fondamentale partire dal concetto di *funzione hash crittografica*. Si tratta di un algoritmo che prende un messaggio di lunghezza variabile e produce un *digest* (o impronta) di lunghezza fissa, con le seguenti proprietà fondamentali [Nat24, Uni24]:

- **Determinismo**: lo stesso input produce sempre lo stesso output.
- **Efficienza**: il calcolo di $H(x)$ è computazionalmente veloce.
- **Resistenza alla preimmagine**: dato y, è computazionalmente impossibile trovare x tale che $H(x) = y$.
- **Resistenza alla seconda preimmagine**: dato x, è impossibile trovare $x' \neq x$ tale che $H(x') = H(x)$.
- **Resistenza alle collisioni**: è impossibile trovare una qualsiasi coppia (x, x') con $x \neq x'$ tale che $H(x) = H(x')$.

Un esempio concreto illustra l'importanza di queste proprietà. Applicando SHA-256:

```
Input:  "Pago Mario 100€"   →   SHA-256: a3f1d8c2...
Input:  "Pago Mario 101€"   →   SHA-256: 7b29e4a1...
```

Cambiare anche un solo carattere nell'input produce un output completamente diverso: questo fenomeno è noto come *effetto valanga*, ed è ciò che rende le funzioni hash utili per rilevare qualsiasi manomissione di un documento.

Strettamente legati a queste funzioni sono i *MAC* (Message Authentication Code), strumenti utilizzati nel contesto della crittografia simmetrica per garantire che un messaggio sia autentico e non sia stato manipolato durante la trasmissione [BR05].

Come funziona una firma digitale classica

Lo schema classico di firma digitale (RSA, ECDSA) si articola in tre fasi. Nella prima fase di **generazione delle chiavi**, il firmatario dispone di una coppia (sk, pk) — chiave privata e pubblica — legate matematicamente: con RSA.

Nella fase di **firma**, il mittente non firma direttamente il messaggio M (che potrebbe essere di dimensione arbitraria), ma il suo digest:

$$M \longrightarrow h = H(M) \longrightarrow \sigma = sk(h)$$

Il digest viene quindi cifrato con la chiave privata, producendo la firma σ.

Nella fase di **verifica**, il destinatario, in possesso della chiave pubblica pk, riceve la coppia (M, σ), calcola autonomamente $h = H(M)$, decifra σ ottenendo h', e accetta la firma come valida se e solo se $h = h'$. La correttezza dello schema si basa sul fatto che solo chi possiede sk può produrre una firma σ tale che $pk(\sigma) = H(M)$, e che la resistenza alle collisioni di H impedisce di trovare un messaggio alternativo M' con lo stesso digest — rendendo impossibile alterare il documento mantenendo la firma valida.

La minaccia quantistica e la strategia del NIST

Come abbiamo ampiamente discusso, RSA e ECDSA sono vulnerabili agli attacchi quantistici, pertanto non sono più considerati sicuri per la firma digitale in un mondo post-quantistico.

Le funzioni hash risultano invece più resistenti alla minaccia quantistica. Il miglior attacco noto in questo contesto è l'algoritmo di Grover, che riduce la

complessità di una ricerca esaustiva da $O(2^n)$ a $O(2^{n/2})$: SHA-256, che offre 256 bit di sicurezza classica, scende a 128 bit di sicurezza quantistica — un livello comunque adeguato ricorrendo a funzioni con output sufficientemente lungo (ad esempio SHA-512, o varianti con 256 bit di sicurezza quantistica).

Nel panorama della crittografia post-quantistica (PQC), affidarsi a protocolli classici come RSA non è dunque più sicuro. D'altra parte, sebbene la crittografia basata sui reticoli rappresenti oggi la soluzione principale, il NIST ha adottato una strategia di *diversificazione* [Nat22]. L'obiettivo è mitigare il rischio che, in futuro, possano essere scoperte vulnerabilità specifiche nei problemi computazionali legati ai reticoli; per questo motivo, è stato standardizzato un protocollo che si basa esclusivamente sulla robustezza delle funzioni hash, senza appoggiarsi ad altri costrutti a chiave pubblica per la firma e la verifica dei messaggi [Nat24].

5.2.1 SLH-DSA (SPHINCS+)

La risposta del nist è 'algoritmo SLH-DSA (standardizzato nel FIPS 205 e basato su SPHINCS+) rappresenta il culmine di decenni di evoluzione nella crittografia basata su hash. Per comprenderlo, è utile ripercorrere i passaggi logici che hanno portato da semplici firme per un singolo bit a una struttura complessa e *stateless* [Nat24].

Firme One-Time: lo schema di Lamport (1979)

Il punto di partenza è lo schema *One-Time Signature* (OTS) di Lamport. L'idea è semplicissima: la sicurezza non deriva da problemi matematici complessi come la fattorizzazione, ma esclusivamente dall'unidirezionalità delle funzioni hash.

Caso base: firmare 1 bit. Il firmatario sceglie due stringhe casuali segrete $x_0, x_1 \in \{0,1\}^n$ e pubblica i loro hash come chiave pubblica:

$$pk = \big(H(x_0),\ H(x_1)\big).$$

Per firmare un bit $b \in \{0,1\}$, rivela semplicemente x_b. Il verificatore controlla che $H(x_b)$ corrisponda al valore corretto in pk. La sicurezza si fonda sull'impossibilità di invertire H: nessun avversario puó ricavare x_0 da $H(x_0)$.

Estensione a un messaggio intero. Per firmare un digest di 256 bit occorrono 256 coppie $(x_{i,0},\ x_{i,1})$, per un totale di 512 segreti:

$$sk = \big\{\ (x_{0,0}, x_{0,1}),\ (x_{1,0}, x_{1,1}),\ \ldots,\ (x_{255,0}, x_{255,1})\ \big\},$$
$$pk = \big\{\ (H(x_{0,0}), H(x_{0,1})),\ \ldots,\ (H(x_{255,0}), H(x_{255,1}))\ \big\}.$$

Per firmare un messaggio M con hash $h = H(M) = b_0 b_1 \cdots b_{255}$, si rivela un valore per ciascun bit:

$$\sigma = (x_{0,b_0},\ x_{1,b_1},\ \ldots,\ x_{255,b_{255}}).$$

Il limite fondamentale di Lamport è che ogni coppia di chiavi può essere usata *una sola volta*. Se si firmano due messaggi diversi con le stesse chiavi, si rivelano

entrambi i segreti $x_{i,0}$ e $x_{i,1}$ per alcuni indici i, consentendo a un avversario di forgiare firme arbitrarie. Inoltre, firme e chiavi pubblica risultano molto grandi (circa 16 KB per un digest di 256 bit) [Nat24, BR05].

Winternitz OTS (W-OTS e WOTS+): catene di hash

Lo schema Winternitz, evolutosi in WOTS+ all'interno di SLH-DSA, riduce drasticamente la dimensione di chiavi e firme rispetto a Lamport, sfruttando il concetto di *catena di hash* [Nat24].

Idea: iterare la funzione hash. Si definisce $f^k(x)$ come H applicata k volte:

$$f^0(x) = x, \qquad f^k(x) = H\big(f^{k-1}(x)\big) \quad (k \geq 1).$$

Con parametro di Winternitz $w = 16$ si lavora con blocchi da 4 bit (valori $v \in [0, 15]$). Per ciascun blocco:

- **Chiave privata**: x (valore casuale),
- **Chiave pubblica**: $f^{15}(x) = H^{15}(x)$ (l'estremo della catena),
- **Firma** del valore v: $f^v(x)$,
- **Verifica**: dato $f^v(x)$ e il valore v, si applica H ancora $15 - v$ volte e si controlla di raggiungere $f^{15}(x)$.

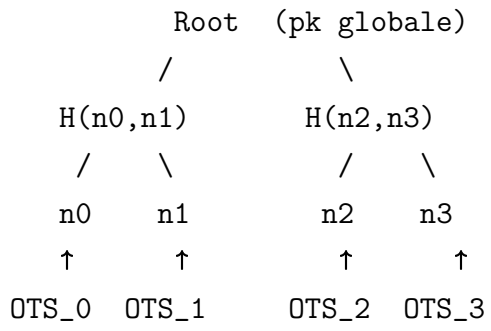
Un avversario che volesse forgiare la firma del valore 3 (più alto) dovrebbe produrre $f^3(x)$: partendo da $f^2(x)$ può solo procedere *avanti* nella catena applicando ulteriori hash, ma non può invertirla. Forgiare un valore *più basso* richiederebbe invece di invertire H, impossibile per ipotesi. Nei sistemi reali si aggiunge un *checksum* che vincola la firma a non poter abbassare i valori, chiudendo entrambe le direzioni di attacco [Nat24].

Albero di Merkle: da One-Time a Many-Time

Lamport e WOTS+ sono schemi *one-time*: ogni chiave può firmare un solo messaggio. Merkle propose di aggregare molte chiavi OTS in un'unica chiave pubblica

tramite un albero di hash, consentendo di firmare un numero arbitrario di messaggi [Nat24, BR05].

Struttura. Si generano 2^h coppie di chiavi OTS. Le chiavi pubbliche vengono messe nelle foglie dell'albero; ogni nodo interno è l'hash della concatenazione dei suoi due figli. La radice è l'unica chiave pubblica globale:



Firma e verifica. Per firmare l'i-esimo messaggio si usa la coppia OTS numero i e si allega il *percorso di autenticazione*, ovvero i nodi fratelli lungo il cammino dalla foglia alla radice. Il verificatore ricalcola la radice partendo dalla firma OTS e dal percorso, confrontandola con la chiave pubblica.

Esempio con 4 foglie (firma con OTS_1).

```
Percorso di autenticazione: [n0,  H(n2,n3)]

Verifica:
  H(pk_OTS_1)           = n1
  H(n0, n1)             = H(n0,n1)
  H(H(n0,n1), H(n2,n3)) = Root
```

Con un albero di $2^{10} = 1024$ foglie è possibile firmare 1024 messaggi con una singola chiave pubblica; il percorso di autenticazione ha lunghezza solo $h = 10$ nodi [Nat24].

XMSS: aggiungere lo stato

XMSS (*eXtended Merkle Signature Scheme*) è l'evoluzione sicura dell'albero di Merkle [Nat24]. Utilizza indirizzi specifici (ADRS) per ogni chiamata alla funzione hash, impedendo attacchi multi-target e garantendo la resistenza post-quantistica.

Il suo principale limite è la natura *stateful*: poiché ogni chiave WOTS+ può essere usata una sola volta, il firmatario deve mantenere un indice che tiene traccia di quali chiavi ha già impiegato. Se tale stato viene perso – ad esempio ripristinando un backup – o riutilizzato per errore, la sicurezza crolla irrimediabilmente.

SLH-DSA

Possiamo quindi analizzare SLH-DSA che è la versione ufficiale di SPHINCS+. La sua innovazione principale è eliminare completamente lo stato, rendendo lo schema *stateless* [Nat24].

Ipertree: un albero di alberi Invece di un unico grande albero di Merkle, SLH-DSA usa una struttura a d livelli in cui ogni foglia di un livello superiore è la radice di un albero al livello inferiore [Nat24]:

```
Livello d-1:  [Albero Merkle 0]
                      |
Livello d-2:  [Albero 0,0]  [Albero 0,1]  ...  [Albero 0, 2^h']
                      |
     ...
Livello 0:    [Albero 0,...,0]  ...  (moltissimi alberi)
                      |
              Foglie FORS  (per il messaggio)
```

Ad esempio, nella variante `SLH-DSA-SHA2-128s` (sicurezza a 128 bit):

$n = 16$ byte, $h = 63$ (altezza totale), $d = 7$ livelli, $h' = h/d = 9$ (altezza del sottoalbero).

Generazione deterministica tramite PRF. Il segreto alla base dello schema stateless è che tutte le chiavi OTS e FORS vengono generate deterministicamente da una chiave segreta master tramite una funzione pseudorandom:

```
sk_seed : 16 byte segreti (la vera chiave privata)
pk_seed : 16 byte pubblici

Per generare la chiave OTS all'indice (layer, tree, leaf):
  x_{layer,tree,leaf} = PRF(sk_seed, ADRS(layer, tree, leaf))
```

dove ADRS è una struttura che codifica l'indirizzo esatto della chiave nell'ipertree. In questo modo non è necessario memorizzare nulla: qualsiasi chiave può essere ricalcolata al volo. Per ogni firma si genera un valore casuale R (o deterministicamente da sk_seed e M), da cui si derivano:

$$(idx\_tree,\ idx\_leaf) = H(R,\ pk\_seed,\ M).$$

Con $h = 63$ l'ipertree conta 2^{63} foglie FORS totali, la probabilità di selezionare due volte lo stesso indice è dell'ordine di $1/2^{63}$, suffucientemente bassa: non è più necessario tenere traccia di alcuno stato.

FORS: Few-Time Signature per il messaggio

Invece di impiegare WOTS+ per firmare direttamente il messaggio finale, SLH-DSA usa FORS (*Forest Of Random Subsets*), uno schema *Few-Time Signature* (FTS) composto da una foresta di k piccoli alberi, ciascuno con 2^a foglie [Nat24].

Il messaggio (dopo hashing) viene diviso in k blocchi da a bit. Ogni blocco è un indice $idx_i \in [0, 2^a - 1]$ che seleziona una foglia in uno dei k alberi:

```
H(messaggio) → k=14 blocchi da a=12 bit ciascuno
             → idx_0, idx_1, ..., idx_13  (ognuno in [0, 4095])

Per ogni albero i:
  - Rivela la foglia idx_i (chiave privata random)
  - Allega il percorso di autenticazione nel sottoalbero i
  - La radice dell'albero i entra nella firma
```

La chiave pubblica di FORS è l'hash delle k radici.

Struttura completa di una firma SLH-DSA

Una firma SLH-DSA è composta da tre elementi principali [Nat24]:

1. **Randomizer** R: valore casuale (o deterministico) che rende il processo non deterministico.
2. **Firma FORS** σ_{FORS}: firma il digest del messaggio tramite la foresta di sottoalberi.
3. **Firma Hypertree** σ_{HT}: una serie di firme WOTS+ che autenticano la chiave pubblica FORS risalendo fino alla radice principale (`PK.root`).

A titolo di confronto, una firma RSA-2048 occupa soli 256 byte. La firma SLH-DSA è significativamente più grande, ma questo è il compromesso inevitabile per garantire sicurezza post-quantistica basata esclusivamente sulle funzioni hash [Nat24, BR05].

Il NIST ha standardizzato 12 varianti con forma: SLH-DSA-hash-nt,

dove `hash` è SHA-2 o SHAKE, n indica il livello di sicurezza (128, 192 o 256 bit) e $t \in \{s, f\}$ distingue la variante *small* (firma compatta, più lenta) da quella *fast* (firma più grande, più veloce). La Tabella 5.1 riporta un confronto tra alcune varianti rappresentative [Nat24].

Table 5.1: Confronto tra varianti SLH-DSA selezionate.

Parametro	**PK**	**SK**	**Firma**
SHA2-128s	32 B	64 B	7 856 B
SHA2-128f	32 B	64 B	17 088 B
SHA2-256s	64 B	128 B	29 792 B

Le chiavi pubblica e privata rimangono compatte; è la firma ad essere grande, perché deve includere tutti i percorsi di autenticazione nell'ipertree e la firma FORS. Questo è il compromesso tipico degli schemi hash-based.

Il filo conduttore dell'intera evoluzione può essere sintetizzato come segue:

$$\text{Lamport OTS} \rightarrow \text{W-OTS} \rightarrow \text{Albero di Merkle} \rightarrow \text{XMSS (stateful)}$$

$$\rightarrow \text{SLH-DSA (stateless).}$$

In ogni passo la sicurezza si basa esclusivamente sulla difficoltà di invertire funzioni hash, senza dipendere da strutture algebriche che l'algoritmo di Shor potrebbe sfruttare [Nat24, BR05].

5.3 Code-based Cryptography

La crittografia basata sui codici rappresenta una delle famiglie più consolidate della crittografia post-quantistica, introdotta da Robert McEliece nel 1978. La sicurezza si basa sul **Syndrome Decoding Problem (SDP)**, un problema NP-hard[9] che consiste nel trovare un vettore di errore e di peso limitato tale che $He^T = s$, data una matrice di controllo H e un vettore sindrome s.

5.3.1 Classic McEliece

Classic McEliece è l'algoritmo più antico tra i candidati PQC[10], con 46 anni di resistenza alla crittanalisi. Utilizza codici di Goppa binari e offre sicurezza conservativa estremamente stabile. Lo svantaggio principale sono le chiavi pubbliche molto grandi (261 KB - 1.3 MB). Non è stato selezionato per la standardizzazione NIST, ma potrebbe essere considerato basandosi sulla futura standardizzazione ISO[11].

5.3.2 HQC (Hamming Quasi-Cyclic)

HQC è stato selezionato dal NIST nel marzo 2025 come quinto algoritmo per la standardizzazione PQC[12], per servire come backup a ML-KEM. La sicurezza si basa sul problema **Quasi-Cyclic Syndrome Decoding (QCSD)**. Utilizza codici quasi-ciclici che permettono chiavi pubbliche molto più piccole (2-4 KB) rispetto a Classic McEliece. È stato scelto per la diversità crittografica, l'analisi di sicurezza stabile e il tasso di fallimento della decodifica ben compreso[13]. Standard finale atteso entro il 2027.

[9] Classic McEliece: conservative code-based cryptography
[10] Classic McEliece: Official Website
[11] NIST IR 8545: Status Report on the Fourth Round
[12] NIST Selects Hamming Quasi-Cyclic (HQC) Algorithm
[13] NIST IR 8545: Status Report on the Fourth Round

5.4 Multivariate Cryptography

La crittografia multivariata si basa sul **Multivariate Quadratic (MQ) Problem**: risolvere sistemi di equazioni polinomiali quadratiche multivariate su campi finiti, un problema NP-hard.

5.4.1 Rainbow

Rainbow era un finalista del Round 3 NIST basato sulla costruzione Unbalanced Oil and Vinegar. Offriva firme molto piccole (66 bytes) e operazioni estremamente efficienti. Tuttavia, nel 2022 Ward Beullens ha pubblicato un attacco che ha recuperato la chiave segreta in circa 53 ore su un laptop standard per i parametri SL1[14]. L'attacco sfrutta proprietà strutturali del polinomio pubblico, portando al ritiro di Rainbow dalla standardizzazione. Questo caso dimostra l'importanza dell'analisi crittografica continua e del processo di standardizzazione NIST.

[14]Breaking Rainbow Takes a Weekend on a Laptop

5.5 Isogeny-based Cryptography

La crittografia basata su isogenie utilizza mappature tra curve ellittiche supersingolari che preservano la struttura del gruppo.

5.5.1 SIKE e il suo fallimento

SIKE era un candidato del Round 4 basato su SIDH[15], con chiavi pubbliche molto piccole (2688 bit a 128-bit di sicurezza quantistica) e perfect forward secrecy.

Nel luglio 2022, Castryck e Decru hanno pubblicato un attacco che ha completamente compromesso SIKE[16], violando tutti i parametri in poche ore su un singolo core CPU (SIKEp434 in 1 ora, SIKEp751 in 21 ore). L'attacco sfrutta i punti ausiliari delle curve ellittiche nelle chiavi pubbliche SIDH. NIST ha rimosso SIKE dalla considerazione nell'agosto 2022.

5.5.2 Lo stato attuale

Altri sistemi basati su isogenie rimangono sicuri: CSIDH (non affetto dagli attacchi Castryck-Decru), SQIsign (schema di firma) e QFESTA (sviluppato da NTT nel 2024, attualmente il più efficiente computazionalmente). Non esistono riduzioni di sicurezza a problemi NP-hard noti, rendendo questa famiglia oggetto di studi cauti. Il caso SIKE dimostra i benefici del processo di standardizzazione NIST nell'identificare vulnerabilità prima della distribuzione.

[15]SIKE Specification
[16]NIST IR 8545: Status Report on the Fourth Round

Chapter 6

Considerazioni finali

6.1 Stato attuale dei computer quantistici

Un computer quantistico reale è un sistema fisico composto da qubit che devono mantenere le loro proprietà quantistiche per tutta la durata del calcolo. Le tecnologie attualmente più diffuse si basano su qubit superconduttori, che operano a temperature estremamente basse, prossime allo zero assoluto, tipicamente dell'ordine di alcune decine di millikelvin, al fine di ridurre il rumore termico e preservare la coerenza quantistica [KKY+19].

Il raggiungimento di tali condizioni richiede l'utilizzo di refrigeratori a diluizione, dispositivi complessi e costosi che rappresentano un primo limite alla scalabilità dei sistemi. Anche minime interferenze ambientali possono causare la *decoerenza*, ovvero la perdita delle proprietà quantistiche dei qubit, compromettendo la correttezza del calcolo.

Un'ulteriore difficoltà riguarda il controllo delle operazioni quantistiche. Le porte devono essere applicate con estrema precisione, poiché errori anche molto piccoli tendono ad accumularsi rapidamente con l'aumentare del numero di qubit e della profondità del circuito.

Per mitigare questo problema si ricorre a tecniche di *correzione d'errore quantistica*. Tuttavia, gli schemi attualmente noti richiedono l'impiego di un numero elevato di qubit fisici per realizzare un singolo qubit logico affidabile, tipicamente dell'ordine di centinaia o migliaia [Pre18].

A causa dei limiti tecnologici attuali, in particolare dei tempi di coerenza ridotti e dell'elevato tasso di errore, l'esecuzione di algoritmi quantistici complessi su larga scala, come quelli necessari per rompere RSA a 2048 bit, non è ancora praticabile. Le stime più ottimistiche suggeriscono che potrebbero essere necessari dai 10 ai 20 anni prima che computer quantistici sufficientemente potenti diventino disponibili [Mos18].

6.2 Stato attuale della crittografia post-quantistica

Il processo di standardizzazione del NIST, iniziato nel 2016, ha portato alla selezione di diversi algoritmi post-quantistici nel 2022 [AAC+22]. Tra questi, CRYSTALS-Kyber per lo scambio di chiavi e CRYSTALS-Dilithium per le firme digitali rappresentano i candidati principali per la sostituzione di RSA ed ECC.

Nonostante la disponibilità di questi standard, la transizione verso la crittografia post-quantistica presenta sfide significative. Molti algoritmi PQC richiedono chiavi più grandi e operazioni più costose rispetto ai loro equivalenti classici, il che può comportare problemi di compatibilità con l'infrastruttura esistente e impatti sulle prestazioni.

Per mitigare i rischi durante la fase di transizione, molte organizzazioni stanno adottando un approccio di *crittografia ibrida*, combinando un algoritmo classico con uno post-quantistico [BBF+19]. Questo approccio garantisce sicurezza anche nel caso in cui uno dei due algoritmi si rivelasse vulnerabile in futuro. Implementazioni di questo tipo sono già state annunciate da grandi aziende tecnologiche come Apple (con il protocollo PQ3) e Google (con il supporto per algoritmi ibridi in Chrome).

6.3 Prospettive future

La minaccia quantistica alla crittografia classica, sebbene non immediata, è sufficientemente concreta da richiedere azioni preventive. Il concetto di "*harvest now, decrypt later*" evidenzia come dati cifrati oggi potrebbero essere vulnerabili in futuro, rendendo urgente la protezione delle informazioni sensibili a lungo termine [Mos18].

L'adozione della crittografia post-quantistica non rappresenta semplicemente

una sostituzione tecnica di algoritmi, ma un processo di trasformazione dell'intera infrastruttura di sicurezza digitale. La collaborazione tra enti di standardizzazione, comunità accademica e industria sarà fondamentale per garantire una transizione efficace e tempestiva verso sistemi crittografici resistenti al calcolo quantistico.

In conclusione, mentre i computer quantistici rimangono una sfida ingegneristica significativa, la crittografia post-quantistica si sta rapidamente evolvendo da teoria accademica a realtà implementativa. La finestra temporale per completare la transizione prima che computer quantistici sufficientemente potenti diventino disponibili è limitata, ma i progressi attuali nella standardizzazione e nell'implementazione di algoritmi PQC offrono motivi di ottimismo per la sicurezza digitale futura.

Bibliography

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, et al. Status report on the third round of the nist post-quantum cryptography standardization process. NIST Interagency Report NISTIR 8413, National Institute of Standards and Technology, 2022.

[BBF+19] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Transitioning to a quantum-resistant public key infrastructure. In *International Conference on Post-Quantum Cryptography*, pages 384–405. Springer, 2019.

[Ber09] Daniel J Bernstein. Introduction to post-quantum cryptography. In *Post-Quantum Cryptography*, pages 1–14. Springer, 2009.

[BR05] Mihir Bellare and Phillip Rogaway. *Introduction to Modern Cryptography*. 2005. Lecture notes, University of California San Diego and University of California Davis.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[IBM24] IBM Quantum. Shor's algorithm. IBM Quantum Learning, 2024. Accesso: 2025.

[KKY+19] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2):021318, 2019.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[LL93] Arjen K Lenstra and Hendrik W Lenstra. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer, 1993.

[Mil85] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 417–426. Springer, 1985.

[Mos18] Michele Mosca. Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security & Privacy*, 16(5):38–41, 2018.

[Nat20] National Institute of Standards and Technology. Recommendation for key management: Part 1 – general. Special Publication 800-57 Part 1 Rev. 5, NIST, 2020.

[Nat22] National Institute of Standards and Technology. NIST IR 8413-upd1: Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. NIST Interagency Report 8413, U.S. Department of Commerce, 2022.

[Nat24] National Institute of Standards and Technology. FIPS 205: Stateless Hash-Based Digital Signature Standard. Federal Information Processing Standard 205, U.S. Department of Commerce, 2024.

[NC10] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition, 2010.

[Pol78] John M Pollard. Monte carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978.

[Pre18] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE, 1994.

[Uni24] Università di Bologna. SHA: Introduzione – slide del corso di sicurezza delle reti. Materiale didattico, Università di Bologna, 2024. Accesso riservato agli studenti iscritti al corso.

Acknowledgements

Optional. Max 1 page.