Analisi comparativa e sistematizzazione di cifrari post-quantum basati su codici casuali

Andrea Bellani

January 12, 2025

Abstract

L'obbiettivo di questo lavoro è l'analisi e il confronto di alcuni crittosistemi post-quantum: i crittosistemi di Alekhnovich e lo schema quasi-ciclico. In particolare, a partire dai crittosistemi basati su reticoli illustreremo i principi da cui Alekhnovich ha tratto ispirazione per la realizzazione dei suoi crittosistemi e come da questi ultimi è possibile ottenere lo schema quasi-ciclico, che getta a sua volta i dettami per il crittosistema *HQC*, attualmente in lizza per la standardizzazione da parte del National Institute of Standards and Technology (NIST).

Il documento organizza inoltre i concetti preliminari di matematica e informatica teorica (affrontati nei corsi della laurea triennale di Ingegneria Informatica presso il Politecnico di Milano) necessari per approcciare i fondamenti della teoria dei codici e i crittosistemi illustrati.

Contents

Chapter 1

Concetti preliminari

1.1 Probabilità [5]

Definition 1.1.1 (Distribuzione di probabilità *uniforme*)**.** Sia:

- $\mathbf{S}$, un generico insieme di valori.

Nella nostra trattazione, diremo che $\mathcal{S} \in \mathbf{S}$ è *estratto casualmente* da $\mathbf{S}$ se la probabilità che $\mathcal{S}$ venga estratto tra tutti gli elementi di $\mathbf{S}$:

- è pari a $\frac{1}{|\mathbf{S}|}$;
- è uguale $\forall \mathcal{S} \in \mathbf{S}$.

1.2 Strutture algebriche [4] [16]

Definition 1.2.1 (Struttura algebrica)**.** Sia:

- $\mathbf{A}$, un generico insieme;
- $\mathbf{\Omega}$, un generico insieme di *leggi di combinazione interne* su $\mathbf{A}$ (generiche funzioni $\mathbf{A}^k \to \mathbf{A}$).

La coppia $(\mathbf{A}, \mathbf{\Omega})$ è detta *struttura algebrica*.

1.2.1 Strutture algebriche notevoli

Definition 1.2.2 (Gruppo)**.** Sia:

- $\mathbf{S}$, un generico insieme di elementi;
- $\cdot$, una funzione $\mathbf{S} \times \mathbf{S} \to \mathbf{S}$ per cui valgono le proprietà di:
  - *identità* : $\exists \mathbb{1} \in \mathbf{S}, \forall\ a \in \mathbf{S}, a \cdot \mathbb{1} = \mathbb{1} \cdot a = a$
  - *chiusura* : $a, b \in \mathbf{S} \Rightarrow\ a \cdot b \in \mathbf{S}$
  - *associatività* : $\forall a, b, c \in \mathbf{S}, a \cdot (b \cdot c) = (a \cdot b) \cdot c$
  - *inverso* : $\forall a \in \mathbf{S}, \exists a^{-1}, a \cdot a^{-1} = a^{-1} \cdot a = \mathbb{1}$

$(\mathbf{S}, \cdot)$ è detto *gruppo*.

Osservazione:

- possiamo anche definire delle "varianti" dei gruppi, ad esempio:
  - un *gruppo commutativo* è un gruppo in cui $\cdot$ rispetta anche la proprietà commutativa ($\forall a, b \in \mathbf{S}, a \cdot b = b \cdot a$);
  - in un *monoide*, $\cdot$ non rispetta la proprietà di inverso;
  - in un *semigruppo*, $\cdot$ né rispetta la proprietà di inverso né ammette quella di identità.

Definition 1.2.3 (Anello)**.** Sia:

- $\mathbf{A}$, un generico insieme di elementi;
- $(\mathbf{A}, +)$, un gruppo commutativo su $\mathbf{A}$;
- $(\mathbf{A}, \cdot)$, un semigruppo su $\mathbf{A}$;

se vale anche la proprietà distributiva (sia "destra" che "sinistra") di $\cdot$ su $+$ allora $(\mathbf{A}, (+, \cdot))$ è detto *anello*.

Osservazione:

- si definisce *ideale* di un anello $(\mathbf{A}, (+, \cdot))$ un sottoinsieme $\mathbf{I}$ di $\mathbf{A}$ tale per cui:
  - $\forall i \in \mathbf{I}, \forall a \in \mathbf{A}, i \cdot a \in \mathbf{I} \wedge a \cdot i \in \mathbf{I}$;
  - $\forall i \in \mathbf{I}, \forall a \in \mathbf{A}, i + a \in \mathbf{I} \wedge a + i \in \mathbf{I}$;

Definition 1.2.4 (Campo)**.** Sia:

- $(\mathbf{A}, (+, \cdot))$, un generico anello;
- $(\mathbf{A} \setminus \{\mathbb{0}\}, (+, \cdot))$, un gruppo commutativo definito sulle stesse operazioni dell'anello considerato e sullo stesso insieme escluso l'elemento neutro di $+$;

$(\mathbf{A}, (+, \cdot))$ insieme a $(\mathbf{A} \setminus \{\mathbb{0}\}, (+, \cdot))$ è detto *campo*.

Osservazione:

- un *campo finito* è un campo in cui $\mathbf{A}$ è un insieme finito. Tipicamente si indicano con la notazione $\mathbb{F}_{\#(\mathbf{A})}$.

1.2.1.1 Definizioni su anelli di polinomi

Definition 1.2.5 (Ideale generato da un polinomio)**.** Sia:

- $(\mathbf{A}, (+, \cdot))$, un generico anello di polinomi;
- p, un polinomio di $\mathbf{A}$;

l'ideale generato da p rispetto a $(\mathbf{A}, (+, \cdot))$ coincide con l'insieme di tutti i polinomi di $\mathbf{A}$ multipli di p (in simboli: (p)).

Osservazione:

- dato un vettore $\mathbf{a} \in \mathbb{F}_q^n$, rappresentiamo con la notazione $a(x)$ il polinomio $\sum_{i=0}^{n-1}(a_i x^i)$.

Definition 1.2.6 (Anello quoziente su un ideale generato da un polinomio)**.** Sia:

- $\mathbb{F}_q[x]$, l'anello dei polinomi aventi coefficienti in $\mathbb{F}_q$;
- $(x^n - 1)$, l'ideale generato da $x^n - 1$ in $\mathbb{F}_q[x]$;

L'*anello quoziente* $\frac{\mathbb{F}_q[x]}{(x^n-1)}$ è l'insieme delle classi di equivalenza dei polinomi di $\mathbb{F}_q[x]$ secondo la relazione di equivalenza: $f(x), g(x) \in \mathbb{F}_q[x]$ sono equivalenti se e solo se $f(x) - g(x)$ è multiplo di $x^n - 1$.

1.3 Definizioni di teoria dei codici *lineari* [9] [11] [3]

Definition 1.3.1 (Codice lineare)**.** Sia:

- $\mathbb{K}^n$, un generico sottospazio di dimensione n (per noi, generalmente, $\mathbb{F}_2^n$);

Un *codice lineare* di dimensione m è un sottospazio vettoriale di dimensione m di $\mathbb{K}^n$.

Osservazioni:

- spesso i codici lineari sono indicati con la notazione "$[n, m]$";
- la matrice le cui righe formano una base del codice lineare è detta *matrice generatrice* del codice lineare (dunque è una matrice $\mathbb{K}^{m \times n}$).

Definition 1.3.2 (Codice duale (o ortogonale))**.** Sia:

- $\mathfrak{C}$, un generico codice lineare di $\mathbb{K}^n$;

si definisce *codice duale* di $\mathfrak{C}$ il codice $\mathfrak{C}^\perp := \{\mathbf{x} \in \mathbb{K}^n | \forall \mathbf{c} \in \mathfrak{C}, \langle \mathbf{x}, \mathbf{c} \rangle = 0\}$.

Osservazioni:

- dalle proprietà degli spazi vettoriali:
  - $dim(\mathfrak{C}) + dim(\mathfrak{C}^\perp) = n$
- $(\mathfrak{C}^\perp)^\perp = \mathfrak{C}$

Definition 1.3.3 (Matrice di parità)**.** Sia:

- $\mathfrak{C}$, un generico codice lineare di $\mathbb{K}^n$;

si definisce *matrice di parità* di $\mathfrak{C}$ la matrice generatrice di $\mathfrak{C}^\perp$.

Osservazioni:

- per le proprietà di $^\perp$, la matrice di parità $\mathfrak{C}^\perp$ è la matrice generatrice di $\mathfrak{C}$;
- la matrice di parità risulta molto utile nelle applicazioni della teoria dei codici in quanto si può dimostrare che un elemento appartiene a $\mathfrak{C}$ se e solo se (dunque potremmo usare questo lemma anche come definizione di matrice di parità) il prodotto riga per colonna con la matrice di parità da l'elemento nullo $\mathbb{K}^n$.

Definition 1.3.4 (Sindrome)**.** Sia:

- $\mathfrak{C}$, un generico codice lineare con matrice di parità $\mathbf{H}$ di $\mathbb{K}^n$;

si definisce *sindrome* di un generico elemento $\mathbf{u} \in \mathbb{K}^n$ il risultato del prodotto riga per colonna tra $\mathbf{H}$ e $\mathbf{u}$.

Osservazioni:

- per quanto detto in precedenza, la sindrome di un elemento è l'elemento nullo se e solo se quell'elemento appartiene al codice considerato;

- si dimostra piuttosto velocemente che due elementi hanno uguale sindrome se e solo se la differenza tra i due appartiene al codice considerato.

Definition 1.3.5 (Peso di un vettore)**.** Si definisce *peso* di un vettore il numero di componenti non nulle del vettore.

Definition 1.3.6 (Distanza di Hamming)**.** Siano:

- $\mathbf{x}$ e $\mathbf{y}$, due generici vettori di ugual dimensione;

si definisce *distanza di Hamming* (in simboli: $d_H(\mathbf{x}, \mathbf{y})$) tra i vettori $\mathbf{x}$ e $\mathbf{y}$:

$$\sum_{i=1}^{n} d_H(x_i, y_i), d_H(a, b) := \begin{cases} 1, a \neq b \\ 0, a = b \end{cases}$$

Definition 1.3.7 (Rotazione di un vettore)**.** Sia:

- $\mathbf{a} := (a_0, \ldots, a_{n-1})$, un generico vettore di dimensione n;

l'operazione di rotazione (destra) di un vettore sul generico vettore $\mathbf{a}$ è definita come (in modo analogo è possibile definire la rotazione sinistra):

$$rot(\mathbf{a}) := (a_{n-1}, a_0, \ldots, a_{n-2})$$

Definition 1.3.8 (Codice ciclico)**.** Sia:

- n, un generico naturale (*lunghezza* del codice);
- $\mathbb{F}_q^n$, lo spazio vettoriale di dimensione n sul generico campo finito $\mathbb{F}_q$;
- l, un generico naturale non nullo (*ordine* del codice);

Un codice ciclico $\mathfrak{C}$ di ordine l è un codice:

i. lineare;

ii. sottoinsieme di $\mathbb{F}_q^n$;

iii. tale per cui se $\mathbf{c} \in \mathfrak{C}$ allora anche $\mathbf{c}$ ruotato l-volte appartiene a $\mathfrak{C}$.

Osservazione:

- è possibile generalizzare il concetto di codice ciclico ai *codici quasi-ciclici*, i quali rilassano la proprietà iii vincolando le rotazioni solo a sotto-vettori contigui dei vettori del codice.

1.4 Definizioni di teoria della computazione [2]

1.4.1 La complessità computazionale e la Macchina di Turing

Per i nostri scopi, la misurazione di complessità di un algoritmo tiene conto delle seguenti semplificazioni:

- considera solo il tempo impiegato e le risorse impiegate dalle esecuzioni (si trascurano *costi collaterali* quali tempi di debugging, risorse ambientali impiegate ecc.);
- si astrae dalla tipologia di solutore dell'algoritmo (si veda il paragrafo sulla Macchina di Turing);

- non è misurata su "singoli input" ma in funzione della loro taglia (ovvero si assume che l'algoritmo impieghi per input di uguale taglia lo stesso quantitativo di tempo e di spazio);
- tra tutte le complessità sugli input di una tale taglia, viene scelta la più grande (*caso pessimo*);
- si considerano di uguale complessità ordini di grandezza il cui limite del rapporto a $+\infty$ è costante (ad esempio, $3n^2$ e $4n^2$, oppure logaritmi aventi basi differenti).

Queste semplificazioni ci portano ad utilizzare le seguenti notazioni per specificare la complessità (ci interessa solo quella temporale) di un algoritmo:

- *O-grande* : $f(n)$ è $O(g(n))$ se esiste un fattore $c \in \mathbb{R}^+$ tale per cui $f(n) \leq c \cdot g(n)$ per n sufficientemente grande;
- Ω-*grande* : $f(n)$ è $\Omega(g(n))$ se esiste un fattore $c \in \mathbb{R}^+$ tale per cui $f(n) \geq c \cdot g(n)$ per n sufficientemente grande;
- Θ-*grande* : $f(n)$ è $\Theta(g(n))$ se esistono due fattori $c_1, c_2 \in \mathbb{R}^+$ tali per cui $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ per n sufficientemente grande.

Ad essere più precisi, queste notazioni non identificano funzioni ma classi di funzioni, dunque sarebbe più corretto dire "$f(n)$ appartiene a $O(g(n))$", ad esempio.

Definition 1.4.1 (Macchina di Turing (definizione "di alto livello"))**.** Si definisce *Macchina di Turing* ("a singolo nastro") un formalismo di calcolo definito da:

- un nastro di input *seekable* (ovvero su cui è possibile scorrere avanti e indietro a piacimento);
- un nastro di output seekable;
- un nastro di memorizzazione seekable;
- un organo di controllo per leggere e scrivere sui nastri.

In buona sostanza, un algoritmo per macchina di Turing indica all'organo di controllo quali celle leggere dai nastri e su quali scrivere.

Definition 1.4.2 (Tesi di Church-Turing)**.** Non esiste un problema *effettivamente calcolabile* (ovvero risolvibile da un qualche formalismo di calcolo) che non sia risolvibile da una macchina di Turing.

Benché non sia mai stata dimostrata, allo stato attuale non è mai stato trovato un problema che contraddica questa tesi. La macchina di Turing rappresenta dunque, allo stato attuale, il più potente (dove per *potenza* intendiamo la capacità di risolvere problemi effettivamente calcolabili) formalismo di calcolo mai definito. Tutti i calcolatori, dunque, si può dimostrare, hanno una potenza di calcolo uguale o inferiore a quella della macchina di Turing. Per questo motivo, quando parleremo di "utilizzatore" oppure "solutore" di un problema, faremo sempre implicitamente riferimento a una macchina di Turing oppure a uno dei numerosi formalismi di calcolo ad essa equivalenti (detti, formalismi *Turing-completi*).

Osserviamo inoltre che la macchina di Turing (o un suo equivalente), è il formalismo che più ha senso identificare come utilizzatore nell'analisi di complessità degli algoritmi (o, che dir si voglia, di "problemi", intesi come algoritmi in grado di risolvere quei problemi). La complessità di un algoritmo, in generale, dipende dal suo utilizzatore e se, per una qualche ragione, il formalismo più efficiente per l'esecuzione di un algoritmo non fosse una macchina di Turing, la macchina di Turing sarebbe sicuramente in grado di emularne l'esecuzione.

1.4.2 I problemi di classe NP e NP-Hard

Definition 1.4.3 (Classe NP)**.** Un problema decisionale (problemi la cui risposta è "sì" oppure "no") rientra nella classe *NP* (*Nondeterministic Polynomial time*) se una macchina di Turing deterministica è in grado di verificare in tempo, al più, polinomiale che un generico input del problema darà risposta "sì" (i problemi di classe *P*, invece, sono quei problemi tali per cui indipendentemente dalla risposta è possibile verificare in tempo polinomiale se un generico input darà risposta "sì" o "no").

Si noti che non per tutti i problemi verificare se un generico input dà una certa risposta equivale a calcolare quella risposta (calcolare un risultato può essere strettamente più complesso di verificarne la correttezza).

Uno dei problemi irrisolti dell'informatica teorica è l'equivalenza tra le classi P ed NP, ovvero non è stato ancora trovato un problema appartenente a P ma non a NP (esistono tuttavia problemi NP che non si sa se siano anche P, trovarne uno non significa dunque che NP≠P).

Definition 1.4.4 (Classe NP-Hard)**.** Un problema decisionale rientra nella classe *NP-Hard* se la sua risoluzione in tempo polinomiale (sia nei casi in cui la risposta è "sì" che nei casi in cui è "no") dimostra che NP=P.

Esistono problemi NP-Hard che sono anche NP (detti *NP-Completi*), altri no. In ogni caso, dal momento che NP=P è un problema ancora aperto, allo stato attuale non è stato trovato un problema NP-Hard risolvibile in tempo polinomiale.

1.5 I problemi di decodifica [6] [14] [19] [7]

Sapere che un problema di decisione è NP-Hard può essere di grande interesse per la creazione di crittosistemi infatti, se si è in grado di ridurre la decifrazione (senza informazioni "private") di un crittosistema alla risoluzione di un problema NP-Hard, considerando il fatto che non sono noti problemi NP-Hard risolvibili in tempo polinomiale, si potrebbe ragionevolmente ritenere il crittosistema robusto agli attacchi dei moderni calcolatori (le cui potenzialità di calcolo rendono praticamente intrattabili i problemi di complessità superiore alla polinomiale). Tuttavia, alcuni problemi su cui si basa la crittografia "tradizionale" (come vedremo) sono risolvibili in tempo polinomiale da calcolatori quantistici, compromettendo così i sistemi che si basano su di loro (ritenuti ragionevolmente "sicuri" prima dell'avvento dei calcolatori quantistici). Lo scopo della crittografia post-quantistica è, in sostanza, quello di realizzare crittosistemi impenetrabili ad algoritmi per calcolatori quantistici e un modo, allo stato attuale, ragionevolmente sicuro per raggiungere questo scopo, è la riduzione ad alcuni problemi NP-Hard.

Definition 1.5.1 (Search decoding problem)**.** Sia:

- k, n tali che $\exists R_1, R_2, 0 < R_1 \leq \frac{k}{n} \leq R_1 < 1$;
- $\mathbf{G} \in \mathbb{K}^{k \times n}$, una generica matrice di rango massimo;
- $\mathbf{e} \in \mathbb{K}^n$, un generico vettore di peso assegnato t;
- $\mathfrak{C}$, il codice lineare la cui matrice generatrice è $\mathbf{G}$;

il *search decoding problem* consiste nel trovare un algoritmo che ricevuto $\mathbf{mG} + \mathbf{e}$ restituisca $\mathbf{m}$.

Definition 1.5.2 (Decisional decoding problem)**.** Sia:

- k, n tali che $\exists R_1, R_2, 0 < R_1 \leq \frac{k}{n} \leq R_1 < 1$;

- $\mathbf{G} \in \mathbb{K}^{k \times n}$, una generica matrice di rango massimo;
- $\mathbf{e} \in \mathbb{K}^n$, un generico vettore di peso assegnato t;
- $\mathfrak{C}$, il codice lineare la cui matrice generatrice è $\mathbf{G}$;

il *decisional decoding problem* consiste nel trovare un algoritmo che data $\mathbf{G}$ sia in grado di distinguere un generico vettore da un vettore $c + e$, dove $c \in \mathfrak{C}$.

Osservazioni:

- viene chiamata *ipotesi sul decisional decoding problem* l'ipotesi sull'inesistenza di una soluzione di complessità polinomiale a questo problema;
- si indica con *syndrome decoding problem* il decisional decoding problem enunciato attraverso la matrice di parità e non attraverso la matrice generatrice e si dimostra che i due enunciati sono equivalenti.

Definition 1.5.3 (Ipotesi di Fisher-Stern [6])**.** L'*ipotesi di Fisher-Stern* afferma che il search decoding problem e il decisional decoding problem hanno la stessa complessità computazionale.

1.6 Definizioni di crittografia

1.6.1 Gli argomenti per ibridi [15]

Immaginiamo un cifrario che cripta ciascuna parola in ingresso convertendola in una parola casuale, non potrebbe esistere un cifrario più sicuro: se la parola viene estratta casualmente, non vi è alcuna relazione tra quest'ultima e la parola iniziale, non vi è dunque modo di invertire il processo di criptazione. Chiaramente, un cifrario di questo tipo non può avere alcuna utilità pratica, in quanto non prevede un processo di decrittazione diverso dall'estrazione casuale. Chiamiamo questo cifrario, *cifrario ideale*.

Un *argomento per ibridi* da un cifrario a un altro è una successione di conversioni che, sotto le ipotesi di sicurezza considerate, dimostra l'equivalenza tra i due. Ad esempio, se un crittosistema cripta una parola x come $x + e$, dove e è estratto casualmente con determinate proprietà, nell'ipotesi che $x+e$ sia indistinguibile da e a meno di non possedere e, si può assumere che il cifrario considerato sia identico al cifrario ideale. In questo caso, l'unica conversione è il passaggio da $x + e$ a un vettore estratto casualmente.

Chapter 2

Considerazioni preliminari sugli algoritmi presentati

Nei capitoli seguenti verranno illustrate, tra le altre cose, le procedure (in pseudocodice) di criptazione e decriptazione dei cifrari analizzati. Per implementare queste procedure è necessario fare uso di passaggi algebrici quali prodotti riga per colonna, prodotti scalari ecc. . Dal momento che confronteremo le complessità di queste procedure, è doveroso specificare a quali complessità facciamo riferimento per le varie operazioni algebriche:

	Complessità temporale
$G \in \mathbb{S}^{n \times m}, H \in \mathbb{S}^{m \times k}, GM$	$O(n \times m \times k)$
$v, w \in \mathbb{S}^{1 \times n}, \langle v, w \rangle$	$O(n)$
$G \in \mathbb{S}^{n \times m}, G^T$	$\Theta(n \times m)$
$G \in \mathbb{S}^{n \times n}, G^{-1}$	$O(n \times n \times n)$

Table 2.1: Complessità temporali delle procedure di interesse

Specifichiamo anche che queste complessità fanno riferimento ad algoritmi non quantistici (nella nostra trattazione solo l'attaccante è un solutore quantistico) e non galattici, in quanto il nostro obbiettivo è individuare crittosistemi di utilità pratica, proprio quello che non sono gli algoritmi galattici.

Inoltre, in diversi algoritmi vengono estratti casualmente vettori o matrici. Modellizziamo questo con le procedure randFromField e randFromCode, le quali rispettivamente estraggono casualmente un elemento da un campo oppure da un codice lineare (di cui viene fornita la matrice generatrice o di parità). Ambedue le procedure hanno come secondo parametro, opzionale, un numero che indica il peso che deve avere l'elemento estratto.

Chapter 3

La crittografia basata su reticoli [12] [10]

Prima di addentrarci nella descrizione dei cifrari analizzati è doveroso illustrare, a grandi linee, il processo che ha portato alla formulazione di questi.

La crittografia pre-quantistica è stata in larga misura basata su problemi matematici computazionalmente "difficili" (algoritmi di complessità non polinomiale), quali fattorizzazione (es. algoritmo RSA) o calcolo di logaritmi discreti (es. algoritmo Diffie-Hellman). Queste tecniche sono così divenute insicure nel momento in cui sono stati inventati algoritmi, per calcolatori quantistici, in grado performare in tempo polinomiale questi calcoli (si veda, ad esempio, l'algoritmo di Shor), almeno a livello teorico. I calcolatori quantistici di cui si dispone allo stato attuale non sono in grado di eseguire con successo questi algoritmi su casi di utilità pratica, ma solo su specifici sotto-casi e comunque di dimensioni molto ridotte rispetto a quelle dei casi di utilità pratica.

Nell'attesa dunque che si realizzino calcolatori quantistici sufficientemente performanti, è necessario sviluppare crittosistemi in grado di resisterli, ma che siano anche abbastanza efficienti da avere un'utilità pratica. Le principali preoccupazioni nella realizzazione di un algoritmo post-quantistico sono, come vedremo più nel dettaglio nel confronto tra i cifrari:

- l'efficienza dei processi di criptazione e decrittazione;
- le dimensioni delle chiavi;
- le prove di correttezza (possono avere luogo errori nel processo di decodifica anche se si è in possesso di tutte le informazioni "private");
- le prove di sicurezza.

Affinché un crittosistema possa avere un'utilità pratica deve bilanciare il trade-off tra questi aspetti (ad esempio, come vedremo, i crittosistemi di Alekhnovich basano la loro sicurezza, praticamente, sugli stessi principi del crittosistema più promettente che vedremo, ma si rivelano molto meno efficienti).

Per arrivare al crittosistema finalista ci è utile partire dalla crittografia basata su reticoli. Negli ultimi decenni è stato dedicato grande interesse all'applicazione di problemi sui reticoli per la crittografia. Esistono, infatti, diversi problemi NP-Completi sui reticoli (ad esempio, "trovare il vettore più vicino a un vettore dato all'interno di un reticolo"), per i quali non sono noti algoritmi quantistici in grado di risolverli in tempo polinomiale. In più, per alcuni di questi problemi, si dimostra che la difficoltà resta analoga tra il caso medio e i casi (magari "pochi") generalmente considerati come "pessimi".

3.1 Il crittosistema di Ajtai-Dwork [8]

Il primo esempio di crittosistema a chiave pubblica la cui sicurezza è ridotta alla difficoltà di un problema definito per i reticoli è stato il crittosistema di Ajtai-Dwork (il quale, tra le altre cose, ha gettato le basi per tutti i successivi schemi basati su reticoli). In buona sostanza, il crittosistema si basa su un problema che si è dimostrato avere nel caso medio la stessa difficoltà di un problema la cui difficoltà era apprezzabile (solo) per il caso pessimo. Si è trattato di un risultato di fondamentale importanza ma, come naturale nei lavori seminali, inadatto a un'implementazione pratica, a causa, ad esempio, delle elevate dimensioni delle chiavi ($O(n^4)$ per la chiave pubblica e $O(n^2)$ per la chiave privata). In breve, questo crittosistema cripta un bit "0" come un elemento estratto casualmente dal reticolo e un bit "1" come una somma di elementi di un sottoinsieme estratto casualmente dal reticolo (anche la criptazione "per bit" anziché "per parola" degrada notevolmente le prestazioni).

3.2 Il crittosistema di Regev [13]

Il crittosistema di Ajtai-Dwork è stato "aggiustato", sotto diversi aspetti, da diversi crittosistemi negli anni seguenti, tra cui il crittosistema di Regev. Quest'ultimo basa la propria sicurezza sul *learning with errors problem* (ideato da Regev a partire dai problemi su cui si basava il crittosistema di Ajtai-Dwork). Lo introduciamo nella formulazione *decisionale* (è quella che ci interessa per introdurre i crittosistemi di Alekhnovich):

Definition 3.2.1 (Decision Learning with errors problem)**.** Siano:

- $q, n, m, \beta \in \mathbb{N}$, con n sufficientemente grande (es. $n \geq 100$) e $\beta \ll q$;
- $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, una generica matrice in $\mathbb{Z}_q^{n \times m}$:

  non è possibile distinguere $\mathbf{As} + \mathbf{e}$ da un vettore estratto casualmente da $\mathbb{Z}_q^n$;

dove:

- $\mathbf{s} \in [-\beta; \beta]^m$, estratto casualmente;
- $\mathbf{e} \in [-\beta; \beta]^n$, estratto casualmente.

In breve, il crittosistema di Regev, così come il crittosistema di Ajtai-Dwork, cripta "bit per bit". Il bit μ viene criptato come:

$$\mathbf{Ax} + \begin{pmatrix} 0 \\ \dots \\ 0 \\ \mu\frac{q}{2} \end{pmatrix}$$

dove:

- $\mathbf{A}$, è la chiave pubblica;
- $\mathbf{x} \in \{0,1\}^m$, un vettore di m bit estratto casualmente;
- q, è lo stesso q visto nella definizione del learning with errors problem.

e decrittato moltiplicando scalarmente il risultato della criptazione per la chiave privata. Questo in realtà non è esatto ma per quello che ci interesserà, ci è più che sufficiente sapere che il processo di criptazione consiste ("semplicemente") in un prodotto scalare con la chiave privata.

Come vedremo anche nei crittosistemi che analizzeremo nel dettaglio, il crittosistema di Regev non garantisce la corretta decodifica ma solo una probabilità ragionevolmente alta per determinati parametri del crittosistema.

Chapter 4

I crittosistemi di Alekhnovich

I *crittosistemi di Alekhnovich* rappresentano il primo esempio di crittosistemi la cui sicurezza si riduce alla risoluzione dei problemi di decodifica (i quali, come visto nella sezione dedicata, sono oramai ben noti per essere computazionalmente difficili da risolvere anche con l'ausilio di calcolatori quantistici). Semplificando molto, durante la procedura di codifica viene inserito nell'informazione "utile" un vettore estratto casualmente (nei paragrafi successivi, indicato con $\mathbf{e}$, o "errore"), il quale rende indistinguibile per un attaccante non in possesso della chiave privata, la decodifica del messaggio iniziale (a meno di risolvere un problema riducibile al decisional decoding problem).

Questi crittosistemi rappresentano, inoltre, le fondamenta dei moderni meccanismi di crittografia basati su codici.

4.1 Primo crittosistema di Alekhnovich [6] [19]

Sia:

- t, un parametro (naturale) assegnato $o(\sqrt{n})$;
- $\mathbf{e}$, un vettore estratto casualmente (da $\mathbb{F}_2^n$) di peso t;
- $\mathfrak{C}$, un codice lineare su $\mathbb{F}_2^n$ avente matrice di parità $\mathbf{H}$, ottenuta apponendo a una generica matrice una riga $\mathbf{xA} + \mathbf{e}$, dove:
  - $\mathbf{x}$ è un generico vettore di $\mathbb{F}_2^k$;
  - $\mathbf{A}$ è una generica matrice conforme alle dimensioni che dovrebbe avere $\mathbf{H}$ per le proprietà dei codici (ovvero $k \times n$);

il primo sistema di Alekhnovic prevede come:

- chiave pubblica : la matrice generatrice di $\mathfrak{C}$ (indicata come $\mathbf{G}$ nelle dimostrazioni);
- chiave privata : il vettore $\mathbf{e}$;
- processo di criptazione: sul singolo bit;
- processo di decriptazione: per un singolo bit.

Algorithm 1: Processo di crittazione del primo crittosistema di Alekhnovich

Input: $b \in \{0,1\}, H \in \{0,1\}^{n \times m}, t \in \mathbb{N}, B \subseteq \{0,1\}^{1 \times n}$ *campo*, $e \in B$
Output: $s \in \mathbb{F}_2^{1 \times n}$

```
if b is 0 then
    s ← randFromCode(H, t) + e
else
    s ← randFromField(F_2^{1×n})
end
return
```

Algorithm 2: Processo di decrittazione del primo crittosistema di Alekhnovich

Input: $s \in \{0,1\}^{1 \times n}, e \in B$
Output: $b \in \{0,1\}$

```
b ← ⟨s, e⟩
return
```

4.2 Secondo crittosistema di Alekhnovich [19]

Sia:

- $\mathbf{M} = \mathbf{XA} + \mathbf{E}$, una generica matrice invertibile tale per cui:
  - $\mathbf{X}$, è un generica matrice $\mathbb{F}_2^{n \times \frac{n}{2}}$;
  - $\mathbf{A}$, è una generica matrice $\mathbb{F}^{\frac{n}{2} \times n}$;
  - $\mathbf{E}$, è una generica matrice $\mathbb{F}^{n \times n}$ avente ciascuna delle righe di peso t;
- $\mathfrak{C}_1 := \{x \in \mathbb{F}_2^n | \Phi(x) \in \mathfrak{C}_0\}$, dove:
  - $\mathfrak{C}_0$, un codice di correzione d'errore avente:
    - lunghezza n;
    - un algoritmo di decodifica avente complessità temporale polinomiale (indicato come *decode* in seguito);
    - dimensione maggiore $\frac{n}{2}$;
  - $\Phi : \mathbb{F}_2^n \to \mathbb{F}_2^n$, l'applicazione lineare definita da $\mathbf{M}$ ($\Phi(x) := \mathbf{M}x$);
- $\mathfrak{C}_2$, il codice lineare avente $\mathbf{A}$ come matrice di parità;
- $\mathfrak{C} := \mathfrak{C}_1 \cap \mathfrak{C}_2$, identifichiamo con:
  - k, la dimensione di $\mathfrak{C}$;
  - $\mathbf{G}$, la sua matrice di parità ($k \times n$, di rango pieno);
- m, la lunghezza di un messaggio (la scegliamo $\frac{k}{2}$);

il secondo sistema di Alekhnovic prevede:

- chiave pubblica : la matrice $\mathbf{G}$;
- chiave privata : la matrice $\mathbf{E}$;
- processo di criptazione : sul messaggio;
- processo di decriptazione : sul messaggio.

Algorithm 3: Processo di crittazione del secondo crittosistema di Alekhnovich

Input: $\{0,1\}^{1\times n} \in msg, F_{half} \subseteq \{0,1\}^{1\times m} campo, t \in \mathbb{N}, F \subseteq \{0,1\}^{1\times n}$ *campo*
Output: $s \in \{0,1\}^{1\times n}$

$x \leftarrow [msg, \text{RANDFROMFIELD}(F_{half}, t)]$
$s \leftarrow xG + \text{RANDFROMFIELD}(F, t)$
return

Algorithm 4: Processo di decrittazione del seocondo crittosistema di Alekhnovich

Input: $E \in \{0,1\}^{n\times n}, s \in \{0,1\}^{1\times n}, decode : \mathfrak{C}_0 \to \mathbb{F}_2^{n\times 1}$
Output: $msg \in \{0,1\}^{1\times n}$

$y \leftarrow Es^T$
$y \leftarrow y^T$
$msg \leftarrow decode(y)M^{-1}G^{-1}$
return

4.3 Dimostrazioni di sicurezza

Possiamo identificare due tipologie differenti di attacco a un crittosistema a chiave pubblica (i nomi sono inventati dall'autore):

- *Third Man Decryption Attack* : attacco finalizzato a decodificare con successo informazioni criptate senza l'utilizzo della chiave privata;
- *Private Key Deduction Attack* : attacco finalizzato al calcolo della chiave privata da quella pubblica;

occorre dunque dimostrare la sicurezza di un crittosistema ad ambedue queste tipologie di attacco.

Nel nostro caso, i crittosistemi da analizzare basano la propria sicurezza sull'inesistenza di algoritmi in grado di risolvere i problemi di decodifica che abbiamo illustrato in tempo polinomiale. Le prove di sicurezza saranno dunque basate su riduzioni, ovvero dimostreremo che un algoritmo in grado di performare con successo un attacco (in tempo polinomiale) non può esistere nell'ipotesi che non esista un algoritmo in grado di risolvere un problema di decodifica "difficile".

Le dimostrazioni il cui paragrafo non ha citazioni sono state ideate dall'autore.

4.3.1 Primo crittosistema di Alekhnovich

4.3.1.1 Riduzione di sicurezza per TMDA [19]

Supponiamo esista, per assurdo, un algoritmo (TMDECRYPT) che data la chiave pubblica G del crittosistema sia in grado di ottenere b da s (dove s è il risultato della criptazione del bit b). Nell'assunzione che valga l'ipotesi sul decoding decisionale TMDECRYPT non è in grado di distinguere il caso in cui l'input s è ottenuto sommando e (secondo la definizione data nel crittosistema) oppure è un vettore estratto casualmente. A causa di come il crittosistema crea la matrice di parità del codice di cui G è la matrice generatrice, TMDECRYPT non è in grado di distinguere G generata casualmente dalla G "di Alekhnovich". Criptiamo allora il bit b scegliendo G random uniforme, otteniamo un assurdo: TMDECRYPT è in grado di decrittare sia il caso in cui $b = 0$ che il caso in cui $b = 1$ anche quando essi sono stati criptati come

vettori totalmente random, cosa chiaramente impossibile (a meno di non scegliere "a caso" la soluzione).

4.3.1.2 Riduzione di sicurezza per PKDA

Supponiamo esista un algoritmo che (in tempo polinomiale) data la matrice di parità **H** del codice costruito dal crittosistema di Alekhnovich possa dedurne **e** (si noti che la chiave pubblica del crittosistema è la matrice generatrice, ma in tempo polinomiale è possibile dedurre **H** da **G**). Se esistesse questo algoritmo (chiamiamolo DISCOVER(H)), sarebbe molto semplice realizzare un algoritmo che in tempo polinomiale risolve il Search Decoding Problem:

Algorithm 5: Riduzione al Search Decoding Problem per PKDA

Input: $G \in \mathbb{K}^{k \times n}\, s = mG + e$
Output: $m \in \{0,1\}^{1 \times k}$
1 $e \leftarrow$ DISCOVER($[G, s]$)
2 $a \leftarrow s - e$
3 $m \leftarrow aG^{-1}$
4 **return**

4.3.2 Secondo crittosistema di Alekhnovich

4.3.2.1 Riduzione di sicurezza per TMDA [19]

Supponiamo esista un algoritmo TMDECRYPT in grado di decrittare con successo senza la conoscenza di E (conoscendo solo le informazioni private). Nell'ipotesi del decoding decisionale, TMDECRYPT non è in grado di distinguere tra M (nella forma stabilita dal crittosistema) da una generica matrice M' di rango massimo. Se ne fosse in grado, infatti, potrebbe distinguere $a + e$ (con a somma random uniforme delle righe di A) ed e random di peso t da un vettore random uniforme, il che contraddice l'ipotesi.

Sia allora $\mathfrak{V}$ un codice $[n, m]$ random e z un n-vettore o random uniforme oppure $r + e$ (con r estratto casualmente da $\mathfrak{V}$). Vogliamo utilizzare $\mathfrak{V}$ per individuare quale di queste due possibilità è z. Inoltre, sia $\mathfrak{U}$ un codice $[n, m]$ random e $\mathfrak{C} := \mathfrak{U} \oplus \mathfrak{V}$ il codice in cui ciascuna parola è ottenuta dallo xor bit-a-bit di due parole di $\mathfrak{U}$ e $\mathfrak{V}$ e di dimensione $2m$ (G la sua matrice generatrice). Da $\mathfrak{C}$ generiamo due codici ($\mathfrak{C} = \mathfrak{C}_1 \cap \mathfrak{C}_2$):

- $\mathfrak{C}_1$: aggiungendo parole random a $\mathfrak{C}$ in modo da ottenere un codice di dimensione $\frac{9n}{10}$;
- $\mathfrak{C}_2$: aggiungendo parole random a $\mathfrak{C}$ in modo da ottenere un codice di dimensione $\frac{n}{2}$.

e sia ϕ la mappa associata alla matrice random M' da $\mathfrak{C}_0$ a $\mathfrak{C}_1$.

Utilizziamo allora TMDECRYPT per ottenere due parole e scegliamone uno a caso **m**. Diamogli allora in input $\mathbf{m}G_{\mathfrak{U}} + z$ (dove $G_{\mathfrak{U}}$ è la matrice generatrice di $\mathfrak{U}$):

- se l'algoritmo è in grado di individuare **m** poniamo $\mathbf{z} = \mathbf{r} + \mathbf{e}$
- altrimenti, poniamo **z** uguale a un vettore random.

Dunque, ogni volta che $\mathbf{z} = \mathbf{r} + \mathbf{e}$, l'input che diamo all'algoritmo ha la stessa identica forma del risultato della criptazione di m, contraddicendo così l'ipotesi sul decoding decisionale (sul codice $\mathfrak{V}$).

4.3.2.2 Riduzione di sicurezza per PKDA

Supponiamo esista un algoritmo che (in tempo polinomiale) data G è in grado di trovare E. Dalle osservazioni in [19], la sicurezza del crittosistema non è influenzata se anche la matrice M è pubblica. Ipotizziamo allora che esista un algoritmo DISCOVER(A,M) permette di trovare E (dove A è la matrice A del crittosistema, che è possibile ottenere da G in tempo polinomiale). Possiamo così scrivere un algoritmo in grado di risolvere il search decoding problem in tempo polinomiale:

Algorithm 6: Riduzione al Search Decoding Problem per PKDA

Input: $G \in \mathbb{K}^{k \times n}, s = mG + e$
Output: $m \in \{0,1\}^{1 \times k}$
1 $E \leftarrow \text{DISCOVER}(G, (1, \ldots, 1)^T s)$
2 $e \leftarrow E_0$ //prima riga di E
3 $a \leftarrow s - e$
4 $m \leftarrow aG^{-1}$
5 **return**

In sostanza, abbiamo usato la stessa strategia vista per il primo crittosistema di Alekhnovich, adattando semplicemente le dimensioni dei vettori.

Chapter 5

Lo schema quasi-ciclico [14]

Sia:

- n, un generico naturale;
- q, una generica potenza prima ($q = p^k$, dove p è generico numero primo e k è un generico naturale);
- $\mathbf{R} := \frac{\mathbb{F}_q[x]}{(x^n-1)}$, l'anello quoziente definito con q ed n;
- $\mathfrak{C}$, un codice lineare $[n, k]$ su $\mathbb{F}_q$ (chiamiamo $\mathbf{G}$ la sua matrice generatrice e, negli algoritmi, *decode* il suo algoritmo di decodifica);
- $\mathfrak{Q}$, un codice quasi ciclico generato casualmente $[2n, n]$ avente matrice di parità costruita come $\mathbf{H} = (\mathbb{1}_{\mathbb{n}\times\mathbb{n}}|rot(\mathbf{h}))$, con $\mathbf{h} \in \mathbb{F}_q^n$;
- $w, w_r, w_e \in [0; \frac{\sqrt{n}}{2}] \in \mathbb{N}$;
- $\mathbf{y}, \mathbf{z} \in \mathbb{F}_q^n$, due vettori di peso di Hamming w;
- $\mathbf{e} \in \mathbb{F}_q^n$, un vettore estratto casualmente di peso di Hamming w_e;
- $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{F}_q^n$, due vettori estratti casualmente di peso di Hamming w_r;

Lo schema quasi-ciclico prevede come:

- chiave pubblica: $(\mathbf{G}, \mathbf{h}, \mathbf{s} := \mathbf{y} + \mathbf{hz}, w_e, w_r)$;
- chiave privata: $(\mathbf{y}, \mathbf{z})$;
- processo di criptazione: sul messaggio;
- processo di decriptazione: del messaggio.

Algorithm 7: Processo di crittazione dello schema quasi-ciclico

Input: $w_r, k, q \in \mathbb{N}, r_1 = \text{RANDFROMFIELD}(F, w_r), r_2 = \text{RANDFROMFIELD}(F, w_r), G \in \mathbb{F}_q^{k \times n}, m, s \in \mathbb{F}_q^{1 \times n}, h \in \mathbb{F}_q^{1 \times n}$

Output: $cpr \in \mathbb{F}_q^{1 \times 2n}$

1 $u \leftarrow r_1 + hr_2$
2 $v \leftarrow mG + sr_1 + e$
3 $cpr \leftarrow [u, v]$
4 **return**

Algorithm 8: Processo di decrittazione dello schema quasi-ciclico

Input: $F \subseteq \{0,1\}^{1 \times n}$ $campo, cpr \in \{0,1\}^{1 \times 2n}, z \in F, decode : \mathfrak{C} \rightarrow \mathbb{F}_q^k$

Output: $msg \in \{0,1\}^{1 \times n}$

1 $msg \leftarrow decode(cpr.secondHalf - cpr.firstHalf\ z)$
2 **return**

Chapter 6

Analogie tra i crittosistemi analizzati

6.1 Analogie tra i crittosistemi basati su reticoli e i crittosistemi di Alekhnovich [1]

I crittosistemi di Alekhnovich, come abbiamo visto, basano la propria sicurezza su un problema (il decisional decoding problem) della teoria dai codici, non della teoria dei reticoli tuttavia, è proprio dai crittosistemi basati su reticoli che Alekhnovich ha tratto ispirazione.

Il primo crittosistema di Alekhnovich è ispirato al crittosistema di Ajtai-Dwork, che non abbiamo analizzato nel dettaglio, ma abbiamo inquadrato i punti fondamentali del crittosistema di Regev (molto simile a quello di Ajtai-Dwork). Potremmo dire che il crittosistema di Regev sta al learning with errors problem (nell'enunciato "decisionale") come il primo crittosistema di Alekhnovich sta al decisional decoding problem:

- Decisional decoding problem: data $\mathbf{G}$, distinguere $\mathbf{mG} + \mathbf{e}$ da $\mathbf{m}$;
- Decisional Learning with errors problem: data $\mathbf{A}$, distinguere $\mathbf{As}+\mathbf{e}$ da un vettore estratto casualmente dal reticolo.

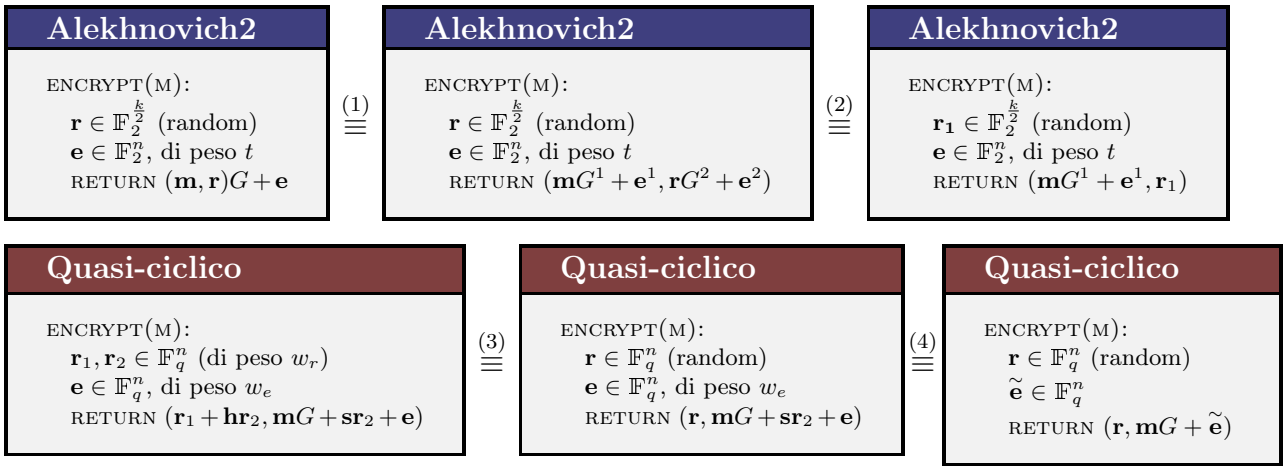
ovvero il Decisional decoding problem "adatta" il Learning with errors problem ai codici lineari, utilizzando anziché un reticolo, un campo finito e ponendo come unico vincolo su $\mathbf{e}$ il peso. Notiamo infatti che se scegliessimo $\mathbf{e}$ nullo entrambi i problemi si risolverebbero come sistemi lineari dove $\mathbf{A}$ e $\mathbf{G}$ sono le matrici dei coefficienti. L'indistinguibilità si ottiene dall'aggiunta dell'errore, meccanismo "ispirato" ai problemi sui reticoli studiati da Ajtai-Dwork.

Il processo di criptazione sfrutta questa indistinguibilità riducendo la distinzione tra un bit "1" e un bit "0" alla risoluzione di questi problemi. Il processo di decrittazione consiste per entrambi i crittosistemi nel prodotto scalare del risultato della criptazione con la chiave privata, il quale sarà ragionevolmente uguale al bit iniziale se si scelgono adeguatamente i parametri del crittosistema (in realtà il primo crittosistema di Alekhnovich garantisce questo solo per i bit "0").

6.2 Dai crittosistemi di Alekhnovich allo schema quasi ciclico [1] [18]

I crittosistemi di Alekhnovich sono stati i primi a fornire dimostrazioni di sicurezza che non si basassero sulla natura dei codici utilizzati. Altri crittosistemi infatti, nascondono (in qualche modo) la struttura del/dei codice/i scelto/i. È stata proprio questa caratteristica a renderli interessanti dal punto di vista teorico per la definizione di crittosistemi più efficienti come lo schema quasi-ciclico.

Nello schema quasi-ciclico si fa uso di due codici. Un codice random lineare (esattamente come i crittosistemi di Alekhnovich) la cui unica caratteristica (dovrebbe) essere quella di avere una decrittazione rapida e un codice random quasi-circolante che produce il consueto "errore" da aggiungere per soddisfare il decisional decoding problem ed esattamente come i crittosistemi di Alekhnovich, entrambi questi codici sono pubblici (ma non del tutto generici a causa della quasi-circolarità). In realtà, tra i due la somiglianza è ancora più grande, mostriamolo applicando un argomento per ibridi ai processi di criptazione del secondo crittosistema di Alekhnovich e dello schema quasi-ciclico (si noti che i processi sono stati riscritti, rispetto alle versioni date in precedenza, in una notazione più comoda all'argomento per ibridi):



(1) possiamo riscrivere l'espressione $(\mathbf{m}, \mathbf{r})G + \mathbf{e}$ come la concatenazione di due componenti, ciascuno dipendente solo da $\mathbf{m}$ o solo da $\mathbf{r}$ ($G = (G^1, G^2)$,$\mathbf{e} = (\mathbf{e}^1, \mathbf{e}^2)$);

(2) sotto la decisional decoding hypothesis, $\mathbf{r}G^2 + \mathbf{e}^2$ è indistinguibile da un vettore random (delle stesse dimensioni);

(3) possiamo assumere che la somma tra due elementi random di peso basso (anche considerando la moltiplicazione per $\mathbf{h}$) sia equivalente a un unico elemento random;

(4) $\mathbf{s}\mathbf{r}_2$ può essere inteso come un vettore di peso basso.

Chapter 7

Analisi comparativa dei cifrari analizzati

7.1 Parametri di confronto dei cifrari

I parametri che si è deciso di utilizzare per confrontare tra loro i cifrari sono:

- parametri legati alle dimensioni:
  - dimensione delle chiavi: dimensioni di chiave pubblica e privata sul numero di bit messaggio;
  - dimensione dei messaggi: rapporto tra numero di simboli di controllo sul numero di simboli di informazione;
- parametri legati alle complessità temporali:
  - complessità del processo di codifica;
  - complessità del processo di decodifica;
- parametri legati alla sicurezza:
  - teoremi a cui è ridotta la loro sicurezza;
- parametri legati alla correttezza:
  - probabilità di decrittare correttamente il contenuto di un messaggio.

Questi rappresentano un corpus di parametri utili a un primo confronto tra crittosistemi, ovvero efficienza, correttezza e sicurezza. La scelta di questi parametri è stata fatta, appunto, tenendo conto di quali sono i parametri che per primi vengono utilizzati per valutare l'affidabilità e l'utilità pratica di un crittosistema. Nella realtà seguono poi analisi più sofisticate in cui si tenta di attaccare il crittosistema con lo scopo di metterne alla luce possibili "falle". Ad esempio, esistono crittosistemi che si dimostrano essere resistenti anche ad attaccanti in possesso di parte delle informazioni private. Oppure, lo scopo di un attacco può non essere necessariamente la decrittazione di un messaggio senza l'ausilio di informazioni private. In un'analisi dettagliata di un crittosistema, infatti, si è interessanti anche alla *Decryption Failure Analysis*, la quale analizza le informazioni ottenibili dagli errori di decodifica del crittosistema. Il risultato di una decodifica errata può infatti dare informazioni interessanti per un attaccante.

7.2 Confronto dei cifrari

7.2.1 Dimensioni delle chiavi

La dimensione delle chiavi di un crittosistema è indubbiamente di primario interesse per la valutazione dell'utilità pratica di un crittosistema. La dimensione delle chiavi, infatti, oltre ad essere (come vedremo) strettamente legata ai tempi di codifica/decodifica di un crittosistema, degrada le prestazioni del processo di scambio delle chiavi e limita fortemente il parametro n, la cui dimensione irrobustisce (esponenzialmente) il crittosistema da parte di attacchi di forza brutta.

Benché nessuno dei crittosistemi analizzati abbia chiavi di dimensioni quali $O(n^4)$ del crittosistema di Ajtai-Dwork, anche $O(n^2)$ è generalmente considerato troppo grande per un crittosistema di utilità pratica.

	Chiave pubblica	Chiave privata
Alekhnovich 1	$\Theta(n \times m)$	$\Theta(n)$
Alekhnovich 2	$\Theta(k \times n)$	$\Theta(n^2)$
Quasi-cyclic	$\Theta(n)$	$\Theta(n)$

Table 7.1: Confronto dimensioni chiavi cifrari

7.2.2 Dimensioni dei messaggi

Su n simboli di informazione:

	Informazioni di controllo su n bit di informazione
Alekhnovich 1	$\Theta(n^2)$
Alekhnovich 2	$\Theta(n)$
Quasi-cyclic	$\Theta(n)$

Table 7.2: Confronto dimensioni dei messaggi

Ad eccezione del primo crittosistema di Alekhnovich, tutti gli altri aggiungono un numero di bit lineare sul numero di bit di informazione, proprio perché la criptazione non avviene "sul singolo bit".

7.2.3 Processi di codifica

	Complessità temporale su n bit da codificare:
Alekhnovich 1	$\Theta(n^2)$
Alekhnovich 2	$\Theta(n^2)$
Quasi-cyclic	$\Theta(n \times k), k \in \mathbb{N}$

Table 7.3: Confronto complessità temporali dei processi di codifica

Notiamo qui un aspetto non ancora affrontato sui crittosistemi. In generale, neanche nello schema quasi-ciclico (che è stato proposto come base per un crittosistema di utilità pratica), la complessità della codifica è un punto di forza. Tra tutti i parametri, infatti, questo è quello generalmente meno considerato nella realizzazione dei crittosistemi, dove si è particolarmente attenti alle dimensioni delle chiavi e ai tempi di decodifica.

7.2.4 Processi di decodifica

	Complessità temporale su n bit da decodificare:
Alekhnovich 1	$\Theta(n^2)$
Alekhnovich 2	$O(n^3)$
Quasi-cyclic	$\Theta(n^{2,00627})$

Table 7.4: Confronto complessità temporali dei processi di decodifica

La complessità di decodifica dello schema quasi-ciclico (dal momento che è strettamente legata alla funzione *decode* scelta) è stata calcolata considerando i dati presentati in [17], i quali sono calcolati su una versione (del 2021) del crittosistema HQC.

7.2.5 Sicurezza [19]

Tutti e tre i crittosistemi analizzati riducono la loro sicurezza al decisional decoding problem (per la precisione, lo schema quasi-ciclico al syndrome decoding problem, ma che si dimostra essere equivalente al decisional decoding problem), che si dimostra essere NP-completo.

7.2.6 Correttezza

Anche disponendo della chiave privata, nessuno degli schemi ha un DFR (*Decoding Failure Rate*) nullo. Analizziamo dunque le probabilità di decodifica corretta dei crittosistemi nel momento in cui si hanno a disposizione tutte le informazioni private (scegliendo i parametri dei crittosistemi secondo quanto è consigliato):

	Probabilità di decodifica corretta di un messaggio
Alekhnovich 1	≈ 1 per i bit "0", 50% per i bit "1"
Alekhnovich 2	≈ 1
Quasi-cyclic	≈ 1

Table 7.5: Confronto affidabilità dei cifrari

List of Tables

Bibliography

[1] M. Alekhnovich. "More on average case vs approximation complexity". In: (2003).

[2] A. Bellani. *Algoritmi e principi dell'Informatica (appunti del corso)*. 2022.

[3] A. Bellani. *Geometria e algebra lineare (appunti del corso)*. 2021.

[4] A. Bellani. *Logica e algebra (appunti del corso)*. 2022.

[5] A. Bellani. *Probabilità e statistica per l'Informatica (appunti del corso)*. 2023.

[6] M. Bombar A. Couvreur T. Debris-Alazard. "On Codes and Learning With Errors Over Function Fields". In: (2022).

[7] M. Bombar A. Couvreur T. Debris-Alazard. "Pseudorandomness of Decoding, Revisited: Adapting OHCP to Code-Based Cryptography". In: (2023).

[8] M. Ajtai C. Dwork. "A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence". In: (1997).

[9] J.T. Gill. *Algebraic Error Correcting Codes (lectures notes)*. 2015.

[10] V. Lyubashevsky. "Basic Lattice Cryptography". In: (2020).

[11] M. Francischello O. Papini. *Teoria dei Codici e Crittografia*. 2012.

[12] C. Peikert. "A Decade of Lattice Cryptography". In: (2016).

[13] O. Regev. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: (2009).

[14] V. Weger N. Gassner J. Rosenthal. "A Survey on Code-based Cryptography". In: (2024).

[15] M. Rosulek. *The Joy of Cryptography*. 2021.

[16] V. Guruswami A. Rudra M. Sudan. *Essential Coding Theory*. 2023.

[17] HQC team. "HQC: Hamming Quasi-Cyclic". In: (2021).

[18] C. Aguilar-Melchor O. Blazy J. Deneuville P. Gaborit G. Zémor. "Efficient Encryption From Random Quasi-Cyclic Codes". In: (2018).

[19] G. Zémor. "Notes on Alekhnovich's cryptosystems". In: (2016).