

UNIVERSITÀ DEGLI STUDI DI CATANIA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

CORSO DI LAUREA IN INFORMATICA

---

ANTONIO NICOLSI

SICUREZZA E APPLICAZIONI  
CRITTOGRAFICHE DELLA FUNZIONE  
DIFFIE-HELLMAN

TESI DI LAUREA

Relatore:  
Chiar.mo Prof. DOMENICO CANTONE

Correlatore:  
Chiar.mo Prof. DAN BONEH

---

Anno Accademico 2000–2001

*A Nelly*

# Indice

<b>Prefazione</b>	<b>vii</b>
<b>1 Crittografia e Sicurezza</b>	<b>1</b>
1.1 Introduzione . . . . .	2
1.2 Crittografia a Chiave Segreta . . . . .	3
1.3 Crittografia a Chiave Pubblica . . . . .	8
1.3.1 Limiti della Crittografia a Chiave Segreta . . . . .	8
1.3.2 Funzioni one-way . . . . .	10
1.3.3 Problemi Computazionali e Funzioni One-Way . . . . .	13

1.3.4	Sicurezza delle Funzioni One-Way . . . . .	17
<b>2</b>	<b>Il Logaritmo Discreto in <math>\mathbb{Z}_p^*</math></b>	<b>19</b>
2.1	Introduzione . . . . .	20
2.2	Algoritmi per il Logaritmo Discreto in $\mathbb{Z}_p^*$ . . . . .	21
2.2.1	L'algoritmo di Shanks . . . . .	22
2.2.2	L'algoritmo di Pohlig-Hellman . . . . .	24
2.2.3	Il metodo dell' <i>Index Calculus</i> . . . . .	28
2.3	Sicurezza del Logaritmo Discreto . . . . .	32
2.3.1	Il Problema delle Informazioni Parziali . . . . .	32
2.3.2	Bit-Security del Logaritmo Discreto . . . . .	36
<b>3</b>	<b>La funzione Diffie-Hellman</b>	<b>43</b>
3.1	Il Problema Diffie-Hellman . . . . .	44
3.2	Assunzioni Computazionali legate al DHP . . . . .	48

3.3	Sicurezza della funzione Diffie-Hellman . . . . .	56
3.3.1	DDH versus CDH . . . . .	58
<b>4</b>	<b>Bit Security della funzione Diffie-Hellman</b>	<b>62</b>
4.1	Il Problema del Numero Nascosto . . . . .	63
4.1.1	Introduzione . . . . .	63
4.1.2	Possibili formulazioni del Problema . . . . .	64
4.1.3	Risoluzione per Approssimazione in un Reticolo . . . . .	67
4.1.4	Applicazione alla Bit Security della funzione DH . . . . .	77
4.2	Sicurezza di bits interni della funzione DH . . . . .	80
4.2.1	Hidden Number Problem per $k$ bits interni . . . . .	81
4.3	Conclusione . . . . .	88
<b>A</b>	<b>Approssimazione nei Reticoli</b>	<b>89</b>
A.1	Spazi Vettoriali . . . . .	90

A.2	Reticoli e Forme Ridotte della Base . . . . .	95
A.3	L'algoritmo LLL . . . . .	98
A.4	Un Problema di Programmazione Intera . . . . .	100
	<b>Bibliografia</b>	<b>101</b>

# Prefazione

Lo sviluppo incessante che la Crittografia ha conosciuto negli ultimi vent'anni è dovuto in gran parte all'introduzione, a metà degli anni '70, di nuove idee da parte di Diffie e Hellman, che portarono alla nascita della moderna Crittografia a chiave pubblica.

Da allora la Crittografia ha smesso di essere esclusivo appannaggio di governi ed eserciti di mezzo mondo, che la usavano per proteggere i propri segreti e penetrare gli archivi degli avversari, per divenire uno strumento a garanzia della privacy di tutti gli utenti della tecnologia informatica e delle telecomunicazioni.

Conseguenza di questa attività in ambito crittografico è stato l'interesse, da parte di molti esponenti della comunità scientifica, alle questioni connesse a tale disciplina, sia per la possibilità di dare applicazione a problemi astratti

fino ad allora considerati fine a se stessi (come la maggior parte dei risultati in *Teoria dei Numeri*), sia per le nuove sfide intellettuali derivanti dalla necessità di fornire una modellizzazione matematica adeguata per uno studio formale delle proprietà di rilievo nelle applicazioni crittografiche.

Una di queste sfide è la formalizzazione del concetto di “sicurezza” per primitive e protocolli crittografici. Ciò perché nello studio della Crittografia non hanno importanza solo le applicazioni rese possibili dalle varie tecniche proposte, ma anche (e soprattutto) la resistenza di tali soluzioni ai possibili attacchi condotti da parte di pirati informatici e professionisti della *Infowar*.

La comprensione di cosa debba intendersi per “primitiva sicura”, e l’analisi di sicurezza delle applicazioni risultanti, costituiscono una condizione essenziale per la diffusione su ampia scala della tecnologia crittografica.

La presente tesi si occuperà, in particolare, di un’interessante primitiva crittografica, la cosiddetta funzione **Diffie-Hellman**, la cui importanza è indicata dal numero di applicazioni che su di essa si basano, e della quale si investigherà la sicurezza, esponendo ciò che è noto, i nuovi risultati ottenuti e i punti oscuri che ancora rimangono da affrontare.

# Capitolo 1

## Crittografia e Sicurezza

*Questo capitolo si prefigge di chiarire cosa si intenda per sicurezza di un'applicazione o di una primitiva crittografica. Dopo aver brevemente considerato le differenze principali fra la Crittografia a Chiave Privata e la Crittografia a Chiave Pubblica, si espongono i presupposti teorici che rendono possibile la Criptazione Asimmetrica e il genere di problemi computazionali sui quali essa si basa. Si evidenzia infine l'importanza che un'attenta analisi delle assunzioni fatte riveste per una adeguata comprensione della sicurezza delle comuni applicazioni crittografiche.*

## 1.1 Introduzione

La Crittografia si occupa dei problemi connessi alla comunicazione fra più entità in presenza di una spia, e delle relative tecniche atte a garantire la sicurezza dell'informazione trasmessa.

Cosa effettivamente si intenda per sicurezza dipende dalla particolare applicazione, poiché sono molteplici le garanzie che i partecipanti alla comunicazione possono richiedere riguardo la comunicazione stessa.

Fra i *desiderata*, il più comune è senza dubbio la *confidenzialità*, ovvero la certezza che solo i legittimi partecipanti alla comunicazione riescano ad interpretarne correttamente il contenuto.

Altrettanto importanti sono spesso l'*autenticazione*, cioè la possibilità di accertare l'identità della persona con la quale si sta intrattenendo la comunicazione (magari senza averla neanche mai incontrata); e l'*integrità dei dati*, ossia la verifica che l'informazione trasmessa, seppur intercettata e letta da parte di un utente malizioso non autorizzato, sia comunque del tutto conforme a ciò che il mittente originario intendeva comunicare.

Ma come garantire tutto ciò nel mondo dei bits, nel quale un'informazione può essere duplicata un qualunque numero di volte, senza che sia possibile

distinguere le varie copie dall'originale, ed ogni singolo bit può essere alterato, senza che ciò comporti alcun segno visibile di effrazione? La Crittografia moderna prende in prestito gli strumenti di svariate discipline matematiche (dall'Algebra alla Teoria dei Numeri, alla Teoria della Complessità), nonché talune tecniche del mondo della Burocrazia (gestione di Certificati, gerarchia di Entità centrali "fidate", etc.), per fornire nell'era digitale soddisfacenti alternative ai tradizionali metodi per la salvaguardia dell'informazione basati sull'apposizione della firma e della ceralacca.

## 1.2 Crittografia a Chiave Segreta

Fra le primitive elementari usate in Crittografia per la progettazione di schemi e protocolli, la più importante è senza dubbio lo *Schema di Criptazione*. Si tratta, nel caso più generale, di una coppia o tripla di algoritmi, che consentono di trasformare un dato messaggio (il *testo in chiaro*) in una forma che risulti inintelligibile a chiunque tranne che al destinatario del messaggio.

Tecniche di codifica segreta assimilabili, in linea di principio, ad uno Schema di Criptazione sono note da più di duemila anni. Ad esempio, Giulio Cesare utilizzava, nelle sue comunicazioni con Roma durante le sue campagne

in Gallia, un semplice artificio che consisteva nel sostituire ogni lettera del testo con il carattere che lo segue di tre posizioni nell'alfabeto latino: quindi ad "ave" corrispondeva "dbh".

È chiaro che un tale schema risulta facilmente interpretabile anche da parte di chi non ne conoscesse il funzionamento; tuttavia esso è interessante perché offre lo spunto per alcune considerazioni generali.

Il difetto principale dello schema sta nella completa mancanza di casualità.

La corrispondenza

testo-in-chiaro  $\longleftrightarrow$  testo-cifrato

è stabilita una volta per tutte, ed è parte integrante dello schema, per cui non è possibile alle due parti della comunicazione (note nella letteratura crittografica con i nomi di Alice e Bob), utilizzare lo stesso schema per parlare con una terza persona senza che anche l'altra parte possa comprenderne senza sforzo il contenuto. Ne segue la necessità di introdurre negli schemi di criptazione un certo grado di incertezza, il che può essere ottenuto facendo dipendere la corrispondenza usata per la sostituzione da una quantità casuale detta *chiave segreta*, nota soltanto ad Alice e Bob. In tal modo risulta più difficile, a chi conosca lo schema usato, ma non la particolare chiave segreta  $k$  adottata, interpretare i testi cifrati.

Gli Schemi di criptazione in cui sia l'operazione di criptazione che quella di decriptazione dipendono dalla stessa quantità  $k$ , vengono detti *Schemi di Criptazione Simmetrici* (o a *Chiave Segreta*, dal momento che la resistenza dello schema alla Crittanalisi<sup>1</sup> dipende dalla segretezza della chiave  $k$ ).

Il fatto più interessante riguardo al *Cifrario di Cesare* è che il principio che sta alla base del suo funzionamento è ancora oggi utilizzato nella quasi totalità dei moderni schemi di criptazione a chiave segreta. Si tratta dell'utilizzo dell'operazione di *sostituzione*, che consiste nel sostituire ogni blocco di testo in chiaro con un blocco corrispondente di testo cifrato.

L'efficacia pratica di tale operazione trova la sua giustificazione nell'analisi teorica dovuto a Shannon [Sha49], e nel lavoro di progettazione e realizzazione di schemi di criptazione efficienti svolto da Feistel [Fei73].

Nella sua investigazione circa i requisiti essenziali per realizzare algoritmi di criptazione in grado di resistere a crittanalisi basata sulla struttura statistica del messaggio da cifrare, Shannon giunse alla conclusione che l'algoritmo di criptazione dovesse presentare due caratteristiche peculiari che egli definì *diffusione* e *confusione*.

---

<sup>1</sup>Per *Crittanalisi* si intende lo studio delle tecniche atte a risalire *sistematicamente* dal testo cifrato al testo in chiaro.

Un algoritmo di criptazione garantisce *diffusione* se ogni parte del testo in chiaro influisce su gran parte del testo cifrato. In tale maniera eventuali “fatti noti” circa il messaggio originale (frequenza dei caratteri, presenza di parole tipiche, etc.) non consentono di dedurre nulla sul cifrato, e viceversa: la diffusione mira quindi a rendere la dipendenza del testo cifrato dal testo in chiaro quanto più complessa possibile.

La *confusione*, invece, riguarda l'altra informazione da cui dipende il testo cifrato, e cioè la chiave segreta: si ha confusione se la relazione fra chiave segreta e testo cifrato è complicata a tal punto da non consentire di risalire alla chiave segreta neanche avendo a disposizione una quantità anche considerevole di coppie testo-in-chiaro  $\longleftrightarrow$  testo-cifrato corrispondenti.

L'importanza di tali proprietà era ben chiara a Feistel, il quale intuì che fosse possibile ottenere entrambe queste preziose caratteristiche ripetendo e alternando le operazioni di *sostituzione* e *permutazione*.<sup>2</sup> In particolare, la sostituzione introduce la dipendenza del cifrato dalla chiave (confusione), mentre la permutazione, cambiando posizione a varie porzioni del testo in chiaro, garantisce che ogni parte del messaggio influenzi il più possibile il testo cifrato

---

<sup>2</sup>L'operazione di permutazione consiste nello scambiare parti del messaggio fra loro, come in un'anagramma.

ottenuto (diffusione). L'utilizzo alternato delle due operazioni, specialmente quando la sostituzione utilizzata nelle varie iterazioni è diversa di volta in volta, complica notevolmente la relazione fra testo in chiaro, testo cifrato e chiave segreta, e al tempo stesso garantisce una conveniente simmetria fra le due operazioni di criptazione e deciptazione.

La robustezza che tale architettura (nota come *Substitution-Permutation Network*, o *SPN*) conferisce agli schemi di criptazione che su di essa si basano, è testimoniata dalla longevità del *Data Encryption Standard*, il crittosistema che, dopo essere stato lo standard per circa venti anni ed aver resistito ad attacchi crittanalitici di ogni tipo, alla fine è stato accantonato perché l'avanzamento tecnologico e l'emergere della computazione distribuita ne hanno reso obsoleti i parametri di sicurezza, che prevedevano chiavi segrete a 56 bits, soggette oggi giorno ad attacchi per ricerca esaustiva.<sup>3</sup>

---

<sup>3</sup>Un attacco per ricerca esaustiva (*brute-force attack*) consiste nel deciptare il testo cifrato provando ad una ad una tutte le possibili chiavi segrete, finché non compaia un testo in chiaro di senso compiuto.

## 1.3 Crittografia a Chiave Pubblica

### 1.3.1 Limiti della Crittografia a Chiave Segreta

Il difetto principale della crittografia simmetrica sta nella necessità, per le due parti che vogliono comunicare in maniera sicura, di stabilire preliminarmente la chiave segreta da utilizzare in seguito. Conseguenza diretta di ciò è che bisogna adoperare tecniche di protezione non crittografiche per la trasmissione della chiave.

Tale tipo di protezione può essere ottenuta mediante incontro di persona (soluzione solitamente sconveniente e poco pratica), o tramite l'uso di reti private e dedicate, il cui traffico non possa essere monitorato da eventuali agenti "maligni" (il che difficilmente può essere garantito per situazioni che coinvolgano anche solo qualche decina di computers).

Un secondo aspetto indesiderabile della crittografia a chiave segreta è che la sicurezza delle varie primitive non sempre è formulata e provata in maniera chiara. Gli schemi di criptazione sopra considerati, ad esempio, si basano per lo più sull'utilizzo di operazioni il cui effetto sul testo in chiaro risulta talmente complesso da farne apparire il risultato del tutto scorrelato dal testo di

partenza; ciò non esclude tuttavia, in linea di principio, l'esistenza di tecniche ingegnose che sfruttino aspetti particolari del crittosistema per interpretare con successo i testi con esso cifrati, pur senza conoscerne la chiave segreta.

Tali ed altri problemi relativi alle tecniche crittografiche convenzionali motivarono la ricerca, da parte di Diffie ed Hellman, di “nuove direzioni”, come recita il titolo del lavoro ad essi dovuto che è unanimemente riconosciuto come l'articolo seminale della *Crittografia a Chiave Pubblica* ([DH76]).

L'idea alla base di questa rivoluzione sta nell'utilizzare due chiavi diverse per le due operazioni di criptazione e deciptazione. Lo scenario tipico non prevede più la presenza di due sole parti nella comunicazione, piuttosto una *rete* di utenti. Ogni partecipante genera una coppia di chiavi, dette rispettivamente *chiave privata* e *chiave pubblica*, e mentre quest'ultima (necessaria alla criptazione dei messaggi), viene resa pubblicamente disponibile a qualunque utente nella rete, la chiave privata viene tenuta segreta, ed utilizzata dal legittimo proprietario per decifrare i messaggi ad esso indirizzati, codificati con la chiave pubblica da chi voglia comunicare con lui in maniera sicura.

Vista la situazione di asimmetria, dovuta al fatto che le operazioni di cifratura e decifratura vengono effettuate adoperando due chiavi diverse, gli schemi di criptazione di questo tipo sono anche detti a *Criptazione Asimmetrica*.

Chiaramente, per garantire la sicurezza del sistema, un requisito minimo è che non sia possibile risalire alla chiave segreta a partire dalla conoscenza dello schema e della chiave pubblica, noti a chiunque.<sup>4</sup>

### 1.3.2 Funzioni one-way

Ma come è possibile che due chiavi distinte consentano di effettuare le operazioni di cifratura e decifratura, che risultino essere l'una l'inversa dell'altra? Tale proprietà può essere ricondotta alla nozione di *funzione one-way con trapdoor*.

Una funzione one-way ('a senso unico') non è altro che una funzione biettiva che risulta 'facile da calcolare' ma 'difficile da invertire'. Per 'facile da calcolare', si intende che esiste un algoritmo polinomiale<sup>5</sup> per valutare la funzione. Di converso, si richiede che la funzione sia 'difficile da invertire' nel senso che qualunque algoritmo, il cui tempo di esecuzione sia limitato da una potenza di  $n$ , fallisce nel tentativo di calcolare la controimmagine su *quasi tutte* le immagini a  $n$  bits della funzione.<sup>6</sup>

---

<sup>4</sup>L'ipotesi che un eventuale agente malizioso conosca tutti i dettagli (eccetto la chiave) del sistema usato è essenziale per costruire schemi di criptazione utilizzabili nella pratica, ed è nota in letteratura con il nome di *Principio di Kerckhoff*.

<sup>5</sup>Un algoritmo è detto *polinomiale* se il tempo che esso impiega quando viene eseguito su un input a  $n$  bits è proporzionale ad una qualche fissata potenza di  $n$ .

<sup>6</sup>Tali nozioni possono essere formulate più rigorosamente nell'ambito della *Teoria della Complessità*, ed in particolare ricorrendo al concetto di *Algoritmo Probabilistico-Polinomiale*, quantificando le opportune probabilità.

Una funzione one-way con trapdoor ('passaggio segreto') è una normale funzione one-way con la caratteristica addizionale che essa risulta facile da invertire a chi sia in possesso di una particolare informazione associata alla funzione stessa, detta per l'appunto informazione trapdoor.

L'informazione trapdoor non deve risultare deducibile dalla descrizione della funzione, poiché altrimenti chiunque potrebbe ricavarla ed utilizzarla per invertire la funzione stessa, che non risulterebbe più essere one-way. Si può pensare all'informazione trapdoor come una sorta di "bacchetta magica" che conferisce, a chi la possiede, la capacità di effettuare dei calcoli che altrimenti non sarebbero (efficientemente) realizzabili: ciò la rende particolarmente preziosa.

Le funzioni one-way possono essere raggruppate in famiglie, in cui le singole funzioni sono individuabili per mezzo di un parametro. Nel caso di una famiglia di funzioni one-way con trapdoor, a ciascun elemento della famiglia corrisponde una diversa informazione trapdoor.

Le funzioni one-way con trapdoor costituiscono il fulcro sul quale si appoggia la Crittografia a Chiave Pubblica, poiché per realizzare la situazione descritta in precedenza, in cui la criptazione e la decrittazione avvengono per mezzo di due chiavi diverse, ma fra loro correlate, è sufficiente scegliere

in maniera casuale una funzione da una famiglia opportunamente costruita di funzioni one-way con trapdoor, distribuire come chiave pubblica una descrizione della funzione selezionata, e tenere come chiave privata la relativa informazione di trapdoor.

I requisiti che una funzione one-way (con o senza trapdoor) deve soddisfare sono talmente straordinari che finora non è stato possibile provarne con certezza matematica l'esistenza. Esistono tuttavia molteplici 'candidate', ovvero funzioni la cui idoneità è subordinata alla fondatezza di talune assunzioni di *Teoria della Complessità* le quali, sebbene sostenute da vari indizi e risultati teorici parziali, e universalmente riconosciute come legittime, non sono però ancora state dimostrate rigorosamente.

Ciononostante, i vantaggi connessi alla Crittografia a Chiave Pubblica sono tali che essa viene comunque utilizzata, adoperando come primitive di base per la realizzazione pratica di tale costruzione teorica le più promettenti 'candidate' funzioni one-way trapdoor, la più diffusa delle quali (nonché la prima ad essere stata proposta) è nota come RSA, dai nomi dei tre ricercatori (Rivest, Shamir e Adleman) che la proposero nel 1978 [RSA78].

### 1.3.3 Problemi Computazionali e Funzioni One-Way

Come già accennato, la Crittografia a Chiave Pubblica è caratterizzata da un approccio più teorico al problema di elaborare tecniche per la codifica dell'informazione, che fa uso di nozioni astratte della *Teoria della Complessità* come ad esempio le funzioni one-way.

La ricerca di funzioni che risultino 'computabili in una sola direzione' ha trovato, nella *Teoria dei Numeri*, una sorgente di problemi interessanti che opportunamente rielaborati consentono di ricavare funzioni one-way.

Si consideri ad esempio il *Problema della Fattorizzazione*:

INPUT:  $m \in \mathbb{N}$

OUTPUT:  $p_1, p_2, \dots, p_r$  primi,  $e_1, e_2, \dots, e_r \in \mathbb{N}$  tali che

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_r^{e_r}$$

Tale problema può essere visto come il problema inverso della moltiplicazione fra interi:

INPUT:  $m_1, m_2, \dots, m_r \in \mathbb{N}$

OUTPUT:  $m = m_1 \cdot m_2 \cdot \dots \cdot m_r$

Esiste una situazione di asimmetria fra questi problemi: mentre è noto un metodo efficiente (cioè che impieghi tempo polinomiale rispetto al numero di bits coinvolti) per eseguire la moltiplicazione fra più interi, non è nota alcuna procedura per fattorizzare un intero  $m$  ad  $n$  bits, il cui tempo di esecuzione sia limitato da un qualche polinomio in  $n$ .

In altri termini, la moltiplicazione fra interi è una funzione one-way, *sotto l'ipotesi* che non sia possibile fattorizzare interi in tempo polinomiale: è bene precisare che quanto detto sopra non esclude che un algoritmo adatto esista; semplicemente si afferma che non è noto in letteratura. Quindi è possibile dire che la funzione in questione è una funzione one-way solo *assumendo vero* un fatto che non è stato provato, per il quale tuttavia, visto lo sforzo fatto dalla comunità scientifica negli ultimi decenni per valutarne la complessità, esistono forti indizi che spingono a considerarlo vero.

Un'idea interessante potrebbe essere quella di cercare di ricavare una funzione one-way con trapdoor, modificando una funzione one-way già nota. Un simile tentativo, basato sul problema della fattorizzazione e sulla relativa funzione one-way, è presente in [RSA78].

Si consideri l'operazione di elevamento a potenza e di estrazione di radici nel gruppo moltiplicativo  $\mathbb{Z}_p^*$ , con  $p$  numero primo.<sup>7</sup> In tale gruppo sono noti algoritmi efficienti per calcolare qualunque potenza. Per quanto riguarda l'estrazione di radici, essa viene ricondotta ad un elevamento a potenza sfruttando la seguente proprietà:

$$(x^a \equiv x^b \pmod{p}) \iff (a \equiv b \pmod{p-1})$$

Alla luce di tale fatto, per calcolare  $\sqrt[e]{x}$  (ovvero quel valore  $y$  per cui risulta che  $y^e \equiv x \pmod{p}$ ), si potrebbe tentare di calcolare il valore  $d$  tale che  $ed \equiv 1 \pmod{p-1}$ , solitamente denotato con  $d \equiv e^{-1} \pmod{p-1}$ . Ciò noto sarebbe possibile ottenere la radice  $e$ -esima di  $x$  semplicemente calcolando  $x^d \pmod{p}$ , poiché:

$$ed \equiv 1 \pmod{p-1} \implies x^d = (y^e)^d \equiv y^{ed} \equiv y \pmod{p}$$

Pertanto estrarre radici in  $\mathbb{Z}_p^*$  è tanto facile quanto calcolare potenze, qualora sia nota la cardinalità del gruppo in questione. In tal caso, infatti, per ottenere l'inverso di un qualunque elemento del gruppo, è sufficiente eseguire il cosiddetto *Algoritmo Esteso di Euclide*.

---

<sup>7</sup>Si tratta dell'insieme dei numeri interi positivi minori di  $p$  e primi con esso, che risulta essere un gruppo rispetto all'operazione di prodotto, ridotta modulo  $p$ .

Nel caso di  $\mathbb{Z}_p^*$  con  $p$  primo, la dimensione del gruppo è sempre nota, in quanto essa è  $p - 1$ . Tuttavia il discorso cambia qualora si consideri il gruppo  $\mathbb{Z}_m^*$ , dove  $m$  è il prodotto di due primi  $p$  e  $q$ . In questo caso il gruppo è costituito da tutti gli interi positivi minori di  $m$  e primi con esso, e la sua cardinalità, denotata con  $\varphi(m)$ , è pari a  $(p - 1)(q - 1)$ . Una proprietà analoga rispetto a quanto visto per  $\mathbb{Z}_p^*$  vale anche in  $\mathbb{Z}_m^*$ :

$$(x^a \equiv x^b \pmod{m}) \iff (a \equiv b \pmod{\varphi(m)})$$

Anche in questo gruppo è possibile calcolare potenze in tempo polinomiale, ma non è più possibile sfruttare lo stratagemma considerato prima per ricondurre l'estrazione di radice alla valutazione di un'opportuna potenza, poiché sarebbe necessario conoscere  $\varphi(m)$ , il che risulta equivalente a fattorizzare  $m$ . Infatti, nota la fattorizzazione di  $m$ , è immediato calcolare  $\varphi(m) = (p - 1)(q - 1)$ .

Viceversa, noti  $\varphi(m)$  e  $m$ , è possibile impostare il seguente sistema:

$$\begin{cases} pq = m \\ (p - 1)(q - 1) = \varphi(m) \end{cases}$$

e quindi, risolvendo rispetto a  $p$  e  $q$ , determinare la fattorizzazione di  $m$ .<sup>8</sup>

---

<sup>8</sup>Si può provare (cfr. ad esempio [Sti95]) che anche il problema di determinare  $d$  (in un qualunque modo) è equivalente a fattorizzare  $m$ .

Pertanto l'operazione di moltiplicazione in  $\mathbb{Z}_m^*$  risulta essere una funzione one-way con trapdoor, in cui l'informazione di trapdoor è la fattorizzazione di  $m$ , sotto l'assunzione che sia difficile estrarre radici nel gruppo. Tale assunzione, nota come *Assunzione RSA*, è diversa dall'ipotesi che fattorizzare interi sia intrattabile, in quanto potrebbe darsi il caso che determinare la fattorizzazione di un dato intero sia in effetti difficile, ma che esista un metodo del tutto scorrelato (e che in particolare non calcoli  $d$ ) per estrarre radici in  $\mathbb{Z}_m^*$ .

### 1.3.4 Sicurezza delle Funzioni One-Way

L'esposizione proposta in precedenza dovrebbe aver fornito un'idea di come vengono ricercate e proposte le 'candidate' funzioni one-way. La situazione è comune a tutte le (presunte) funzioni one-way (con o senza trapdoor) note e in uso nella pratica: esistono problemi computazionali (solitamente nell'ambito della *Teoria dei Numeri*, ma anche della *Teoria dei Codici* e dell'*Ottimizzazione Combinatoria*), per i quali non è noto alcun metodo efficiente per risolverli, ma il cui problema inverso risulta essere invece facilmente affrontabile. Le *assunzioni computazionali* sulle quali le varie funzioni one-way si basano sono spesso fra di loro collegate, tuttavia distinte, e se in effetti è plausibile che alcune di esse siano corrette, per altre la situazione è molto più incerta.

In definitiva, il tentativo di fondare la cifratura dei dati su una base teorica più solida rispetto a quanto avviene nel caso della Crittografia a Chiave Segreta, si può considerare solo parzialmente riuscito, fintantoché non risulti definitivamente accertata la veridicità di questa o quella assunzione computazionale.

In mancanza di ciò è importante determinare le relazioni esatte fra le varie assunzioni, nel tentativo di ridurre al minimo l'incertezza sottostante all'intera costruzione, e di stabilire in quali particolari circostanze risulti legittimo considerare intrattabili i vari problemi.

Per tale ragione è sensato ritenere che ogni avanzamento dello stato di conoscenza riguardo cosa viene assunto, e cosa invece è possibile provare, costituisca un passo avanti verso la piena comprensione della robustezza e dei limiti della tecnologia crittografica che viene comunemente usata.

## Capitolo 2

# Il Logaritmo Discreto in $\mathbb{Z}_p^*$

*Il problema del Logaritmo Discreto è un problema ben noto nella Teoria dei Numeri, per il quale non si conoscono procedure risolutive efficienti, quando i suoi parametri vengono selezionati con cura. In questo capitolo si considera la formulazione di tale problema all'interno del gruppo  $\mathbb{Z}_p^*$  (con  $p$  primo), che ne rappresenta la variante più diffusa nelle applicazioni pratiche. Dopo aver esaminato le tecniche note per il calcolo del Logaritmo Discreto in  $\mathbb{Z}_p^*$ , se ne discutono in dettaglio le proprietà di sicurezza crittografica.*

## 2.1 Introduzione

L'insieme dei numeri interi positivi minori di un primo  $p$  risulta essere un gruppo rispetto all'operazione di moltiplicazione modulare. Questo gruppo, denotato con  $\mathbb{Z}_p^*$ , è anche ciclico, ovvero esiste un elemento  $\alpha \in \mathbb{Z}_p^*$  tale che:

$$\begin{aligned}\mathbb{Z}_p^* &= \{1, \alpha, \alpha^2, \dots, \alpha^{(p-2)}\} \\ &= \{\alpha^i \mid i \in \mathbb{Z}_{p-1}\}\end{aligned}$$

dove tutte le esponenziazioni si intendono ridotte mod  $p$ .

Un tale elemento  $\alpha$  è detto *generatore* del gruppo, ed esso, per come è definito, possiede una caratteristica speciale, per la quale ogni elemento  $\beta \in \mathbb{Z}_p^*$  può essere scritto come una sua potenza:

$$\forall \beta \in \mathbb{Z}_p^* \quad \exists b \in \mathbb{Z}_{p-1} : \beta = \alpha^b \pmod{p}$$

Ricalcando la notazione relativa al campo dei numeri reali  $\mathbb{R}$ , l'esponente  $b$  da dare ad  $\alpha$  per ottenere  $\beta$  è detto *logaritmo (discreto) di  $\beta$  in base  $\alpha$  (rispetto al modulo  $p$ )*. Solitamente, i valori di  $\alpha$  e  $p$  risultano chiari dal contesto, e  $b$  viene denotato con  $\text{Dlog } \beta$ .

Come già accennato parlando della funzione one-way con trapdoor basata sull'assunzione *RSA*, esistono algoritmi efficienti per calcolare qualunque

potenza.<sup>1</sup> Tuttavia, i metodi noti per il calcolo dell'operazione inversa dell'esponenziazione, ovvero il logaritmo discreto, risultano essere troppo lenti nella pratica, se  $p$  viene scelto molto grande e con caratteristiche opportune.

Sono quindi presenti le condizioni per considerare la funzione di esponenziazione modulare come una funzione one-way. Si noti che anche in questo caso tale affermazione è subordinata ad un'ipotesi, vale a dire l'intrattabilità del problema del logaritmo discreto (*Assunzione Dlog*).

L'assunzione Dlog è, fra le ipotesi computazionali utilizzate in Crittografia, una delle più plausibili, ed è ritenuta più "sicura" dell'ipotesi sulla Fattorizzazione, sebbene non siano noti risultati in *Teoria della Complessità* che leghino fra loro le due assunzioni.<sup>2</sup>

## 2.2 Algoritmi per il Logaritmo Discreto in $\mathbb{Z}_p^*$

Si presenta adesso una breve carrellata dei vari algoritmi (inefficienti) noti in letteratura per il calcolo del logaritmo discreto in  $\mathbb{Z}_p^*$ . In tutti i casi,  $\alpha$  e  $p$  si assumono fissati, ed il problema consiste nel ricavare, noto  $\beta$ , il valore  $\text{Dlog } \beta$ .

---

<sup>1</sup>Nel caso dell'operazione di esponenziazione, è possibile trarre vantaggio dal fatto che la base è costante utilizzando delle opportune tabelle precomputeate, velocizzando così i calcoli.

<sup>2</sup>Tale giudizio deriva dal fatto che il migliore algoritmo noto per fattorizzare interi risulta, per quanto inefficiente, comunque più veloce del migliore algoritmo noto per calcolare logaritmi discreti (quando i parametri vengono fissati opportunamente).

### 2.2.1 L'algoritmo di Shanks

Una tecnica banale per calcolare  $\text{Dlog } \beta$  consiste nel calcolare via via tutte le potenze di  $\alpha$ , fino a trovare il valore  $\beta$ . Tale soluzione prende tempo  $\mathcal{O}(p)$  (tralasciando il tempo necessario per effettuare le singole moltiplicazioni), il che risulta esponenziale sulla dimensione del problema.<sup>3</sup>

Un'altro approccio diretto si basa sull'uso di una tabella precomputata, contenente tutte le potenze di  $\alpha$ , riordinata per risultato. Quando è necessario valutare il logaritmo discreto su un valore  $\beta$ , si effettua semplicemente un *look-up* nella tabella. Questo algoritmo ha quindi un tempo di esecuzione costante, a fronte di un preprocessing che richiede tempo e spazio dell'ordine di  $p$  (trascurando anche qui i fattori logaritmici dovuti alle operazioni modulari e all'ordinamento della tabella).

Un metodo che tenta di mediare fra i due casi estremi sopra proposti è dovuto a Shanks. In sostanza, piuttosto che calcolare, in fase di preprocessing, l'intera tabella, se ne costruisce una più piccola (anch'essa ordinata in base al valore) che contiene le potenze di  $\alpha$  con esponente multiplo di  $m \doteq \lceil \sqrt{p-1} \rceil$ .

---

<sup>3</sup>Nel caso in esame, la dimensione del problema è il numero di bits necessari per rappresentare  $\beta$ , che è dell'ordine di  $\log p$ .

Successivamente, in fase di esecuzione, viene calcolata un'analogha tabella, anch'essa con  $\mathcal{O}(m)$  entries, e dal confronto fra le due tabelle viene determinato il valore  $\text{Dlog } \beta$ .

Più in dettaglio, la tabella costruita nella fase di preprocessing contiene tutte le coppie  $(j, \alpha^{mj} \pmod p)$ , al variare di  $j = 0, \dots, m$ . In fase di esecuzione, invece, vengono calcolati i valori  $(i, \beta\alpha^{-i} \pmod p)$ , inseriti in ordine rispetto alla seconda componente in un'altra tabella.

A questo punto, si scorrono simultaneamente le due tabelle (in tempo al più  $\mathcal{O}(m)$ ), alla ricerca di due entries le cui seconde componenti coincidano. Si denotino con  $(j, y)$  e  $(i, y)$  tali entries. Risulta allora:

$$y = \alpha^{mj} \pmod p$$

$$y = \beta\alpha^{-i} \pmod p$$

e dal confronto delle due eguaglianze, si ottiene:

$$\alpha^{mj} \equiv \beta\alpha^{-i} \pmod p$$

$$\beta = \alpha^{mj+i} \pmod p$$

$$\text{Dlog } \beta = mj + i$$

Tralasciando i fattori logaritmici dovuti al calcolo degli esponenziali e all'ordinamento delle tabelle, l'algoritmo di Shanks richiede tempo di preprocessing  $\mathcal{O}(m)$ , spazio di memorizzazione  $\mathcal{O}(m)$  e tempo di esecuzione  $\mathcal{O}(m)$ .

Tale algoritmo, benché di complessità esponenziale, è interessante per la situazione di compromesso più equilibrato che raggiunge in termini delle risorse di spazio e tempo. Esso inoltre è applicabile per il calcolo del Dlog non solo in  $\mathbb{Z}_p^*$ , ma in un qualunque gruppo finito, e risulta essere l'algoritmo generico più efficiente per determinare logaritmi discreti.

### 2.2.2 L'algoritmo di Pohlig-Hellman

Si tratta di un algoritmo per il calcolo del Logaritmo Discreto applicabile quando la fattorizzazione dell'ordine del gruppo (che nel caso di  $\mathbb{Z}_p^*$  è  $p - 1$ ) risulta nota e costituita soltanto da fattori primi relativamente piccoli. Sotto tali ipotesi, risulta possibile calcolare il valore di  $b = \text{Dlog } \beta$  modulo le varie potenze dei fattori primi di  $p$ , e ricombinare alla fine i valori ottenuti per mezzo del *Teorema Cinese del Resto*.<sup>4</sup>

Si supponga pertanto che risulti:

$$p - 1 = \prod_{i=1}^h q_i^{c_i}$$

con  $q_i$  fattori primi distinti. Si consideri un generico fattore primo  $q$ , e sia  $c$  la potenza con la quale esso figura nella formula precedente. Allora  $c$  è la

---

<sup>4</sup>Il *Teorema Cinese del Resto* consente di risolvere efficientemente un sistema di congruenze lineari in cui i moduli risultino a due a due coprimi.

massima potenza per cui risulta che  $q^c$  divide  $p - 1$ , ovvero:

$$p - 1 \equiv 0 \pmod{q^c} \quad \text{ma} \quad p - 1 \not\equiv 0 \pmod{q^{c+1}}$$

Per calcolare il valore  $x \doteq b \pmod{q^c}$ , si esprime l'incognita  $x$  in base  $q$ , e si ricavano via via tutti coefficienti  $x_i$ :

$$x = \sum_{i=0}^{c-1} x_i q^i$$

$$x = b \pmod{q^c} \implies \exists x_c : \quad b = x + x_c q^c$$

$$b = \sum_{i=0}^c x_i q^i$$

Per determinare  $x_0$ , si osservi che:

$$\beta^{(p-1)/q} = (\alpha^b)^{(p-1)/q} = \alpha^{(p-1)b/q} \equiv \alpha^{(p-1)x_0/q} \pmod{p}$$

La congruenza che figura nell'espressione precedente vale poiché una simile relazione sussiste fra gli esponenti. Infatti:

$$\begin{aligned} (p-1) \frac{b}{q} &= (p-1) \frac{\sum_{i=0}^c x_i q^i}{q} \\ &= (p-1) \frac{x_0 + q \sum_{i=1}^c x_i q^{i-1}}{q} \\ &= (p-1) \frac{x_0}{q} + (p-1) \sum_{i=1}^c x_i q^{i-1} \\ &\equiv (p-1) \frac{x_0}{q} \pmod{p-1} \end{aligned}$$

In virtù di tale relazione, per determinare  $x_0$  basta generare via via tutte le potenze del tipo  $\alpha^{(p-1)i/q} \pmod p$ , per  $i = 0, \dots, q-1$ : il valore  $\beta^{(p-1)/q} \pmod p$  si otterrà in corrispondenza di  $i = x_0$ .

Per quanto riguarda il calcolo degli altri coefficienti, è possibile ricondursi alla situazione precedente considerando:

$$\begin{aligned}\beta_1 &= \beta\alpha^{-x_0} \\ b_1 &= \text{Dlog } \beta_1 \\ x' &= b_1 \pmod{q^c}\end{aligned}$$

Risulta allora:

$$x' = \sum_{i=1}^c x_i q^i$$

e quindi (eseguendo dei calcoli analoghi a quelli visti in precedenza):

$$\beta_1^{(p-1)/q^2} \equiv \alpha^{(p-1)x_1/q} \pmod p$$

Calcolando le potenze  $\alpha^{(p-1)i/q} \pmod p$  per  $i = 0, \dots, q-1$  finché non si ottiene  $\beta_1^{(p-1)/q^2} \pmod p$ , si ricava il valore di  $x_1$ , e iterando la procedura per  $x_i, i > 1$  si determina  $x$ .

Per quanto riguarda il costo computazionale di tale procedura, si noti che le potenze  $\alpha^{(p-1)i/q} \pmod p$  per  $i = 0, \dots, q-1$  possono essere calcolate una volta per tutte, e inserite in un'apposita tabella ordinata secondo il valore;

essendo la costruzione della tabella l'operazione più onerosa dell'algoritmo, il tempo necessario per ricavare per intero  $x$  risulta sostanzialmente identico al tempo necessario al calcolo di una sua singola cifra in base  $q$ .

Di conseguenza, utilizzando l'algoritmo di Pohlig-Hellman, calcolare logaritmi modulo  $q^c + 1$  e modulo  $q^{\tilde{c}} + 1$  (con  $\tilde{c} \gg c$ ) comporta essenzialmente la stessa difficoltà computazionale, per quanto i due moduli possano differire per svariati ordini di grandezza: ciò spiega perché l'algoritmo di Pohlig-Hellman risulta efficiente quando i fattori di  $p - 1$  sono tutti relativamente piccoli.

Ripetendo il ragionamento per tutti i fattori primi di  $p - 1$ , ed applicando il *Teorema Cinese del Resto* ai risultati parziali ottenuti, si ottiene il  $\text{Dlog } \beta$ .

L'importanza pratica di tale algoritmo deriva dal fatto che esso mostra come non sia sufficiente scegliere un primo  $p$  molto grande per ottenere un gruppo  $\mathbb{Z}_p^*$  nel quale il problema del logaritmo discreto risulti (probabilmente) intrattabile, ma bisogna scegliere<sup>5</sup> un valore per cui l'ordine del gruppo risulti avere almeno un fattore primo di dimensione paragonabile a  $|p|$ .

---

<sup>5</sup>Un accorgimento molto utilizzato nella pratica è quello di considerare numeri primi del tipo  $p = 2q + 1$ , con  $q$  numero primo. Un tale numero  $p$  viene detto *safe prime*.

### 2.2.3 Il metodo dell'*Index Calculus*

Il metodo dell'*Index Calculus* è l'algoritmo più efficiente<sup>6</sup> noto in letteratura per il calcolo dei logaritmi discreti in  $\mathbb{Z}_p^*$ .

Di tale algoritmo sono presenti più di una variante, poiché esso fa uso di due procedure esterne che possono essere ispirate ad euristiche diverse.

La prima procedura fornisce un'insieme  $\mathcal{B}$  di numeri primi relativamente piccoli (detta *base di fattori*), con la proprietà che una porzione significativa degli elementi del gruppo  $\mathbb{Z}_p^*$  risultano avere quali fattori primi soltanto elementi in  $\mathcal{B}$ . La seconda procedura, invece, fornisce un metodo efficiente per fattorizzare gli elementi di  $\mathbb{Z}_p^*$  che risultino effettivamente esprimibili a partire dai soli numeri primi presenti nella base  $\mathcal{B}$ .

Il metodo dell'*Index Calculus* prevede una fase di preprocessing, il cui scopo è di calcolare il logaritmo discreto modulo  $p$  di tutti i primi della base  $\mathcal{B}$ .

---

<sup>6</sup>Si noti che esiste una classe di gruppi ciclici in cui il problema del logaritmo discreto risulta "difficile", ma nei quali non è possibile adoperare tale metodo (ad esempio, nei gruppi basati su *Curve Ellittiche*).

### Preprocessing

1. Si scelga una *base di fattori*  $\mathcal{B} \doteq \{p_1, \dots, p_B\}$  (Procedura 1);
2. Si scelga in maniera casuale un  $x_i \in \mathbb{Z}_{p-1} \setminus \{0\}$ , finché non risulti che  $\alpha^{x_i}$  sia fattorizzabile in termini dei primi in  $\mathcal{B}$  (Procedura 2):

$$\alpha^{x_i} \equiv p_1^{a_{1i}} \dots p_B^{a_{Bi}} \pmod{p} \quad \iff$$

$$x_i \equiv a_{1i} \text{Dlog } p_1 + \dots + a_{Bi} \text{Dlog } p_B \pmod{p-1}$$

3. Si ripeta il passo precedente  $B$  volte;<sup>7</sup>
4. Si risolva, nelle incognite  $\text{Dlog } p_1, \dots, \text{Dlog } p_B$ , il sistema ottenuto combinando le congruenze lineari introdotte nei punti precedenti.

La fase di calcolo vera e propria del metodo dell'*Index Calculus* consiste di un Algoritmo Probabilistico<sup>8</sup> di tipo *Las Vegas*. Si tratta di una categoria di Algoritmi Probabilistici che possono fornire 2 tipi di output:

- La risposta corretta (con una certa probabilità  $t$ );
- Nessuna risposta (con probabilità  $1 - t$ ).

---

<sup>7</sup>In effetti, solitamente vengono calcolati  $B + c$  valori  $x_i$ , per aumentare la probabilità che il sistema di congruenze lineari che ne deriva ammetta soluzione unica. Il valore della costante  $c$  tipicamente è piccolo, ad esempio  $c = 10$ .

<sup>8</sup>Un Algoritmo si dice *Probabilistico* se, nel corso della computazione, effettua delle scelte casuali.

L'algoritmo in esame tenta di ricondurre il calcolo di  $\text{Dlog } \beta$  ad un'opportuna combinazione lineare dei logaritmi discreti dei fattori della base, procedendo come segue:

### Calcolo del $\text{Dlog } \beta$

1. Si scelga in maniera casuale un  $s \in \mathbb{Z}_{p-1} \setminus \{0\}$ ;
2. Si verifichi (tramite la Procedura 2) se  $\beta\alpha^s \pmod p$  è esprimibile per mezzo dei soli fattori della base  $\mathcal{B}$ ;
3. Se così non è, l'algoritmo termina senza fornire alcuna risposta;
4. Altrimenti, sia:

$$\beta\alpha^s \equiv p_1^{c_1} \dots p_B^{c_B} \pmod p$$

Passando ai logaritmi discreti in ambo i membri, si ottiene

$$\text{Dlog } \beta + s \equiv c_1 \text{Dlog } p_1 + \dots + c_B \text{Dlog } p_B \pmod{p-1}$$

$$\text{Dlog } \beta \equiv c_1 \text{Dlog } p_1 + \dots + c_B \text{Dlog } p_B - s \pmod{p-1}$$

Quest'ultima espressione consente di ricavare il valore  $\text{Dlog } \beta$  cercato, essendo note tutte le altre quantità presenti.

La probabilità che tale Algoritmo *Las Vegas* fornisca una risposta dipende dalla “qualità” della *base di fattori*  $\mathcal{B}$ . Tanto maggiore è la porzione di  $\mathbb{Z}_p^*$  che risulta “coperta” da tale famiglia di numeri primi, tanto più elevata sarà la probabilità che il valore  $s$ , scelto a caso al punto 1, verifichi la condizione del punto 2.

Come già accennato, è possibile considerare molteplici varianti del metodo proposto, a seconda dei criteri-guida adoperati dai due moduli esterni utilizzati nell’algoritmo. Tuttavia, nel caso in questione (cioè del Dlog in  $\mathbb{Z}_p^*$ ), l’ovvia scelta  $\mathcal{B} \doteq \{i \text{ } B \text{ numeri primi iniziali}\}$ , presenta il vantaggio di rendere semplice la determinazione della fattorizzazione degli elementi  $x \in \mathbb{Z}_p^*$  in termini della base  $\mathcal{B}$  (poiché, trattandosi di primi piccoli, è possibile dividere  $x$  per i vari  $p_i$  per stabilirne la fattorizzazione), garantendo comunque una buona copertura del gruppo  $\mathbb{Z}_p^*$ .

È interessante osservare, per ultimo, che tale metodo fa uso delle stesse idee alla base dei migliori algoritmi noti per la fattorizzazione di interi (quali ad esempio l’algoritmo di Dixon o di Coppersmith; per una trattazione più estesa si veda [MvOV96]).

## 2.3 Sicurezza del Logaritmo Discreto

### 2.3.1 Il Problema delle Informazioni Parziali

Dal momento che tutti gli algoritmi esposti in precedenza richiedono tempo più che polinomiale (quando i parametri del problema vengono fissati opportunamente), la valutazione del Dlog risulta un compito al di là delle capacità di calcolo di un qualunque crittanalista reale.

Il fatto che non sia possibile determinare il valore del logaritmo discreto di un dato  $\beta$  non esclude tuttavia, in linea di principio, che si possa agevolmente calcolare qualche informazione circa  $\text{Dlog } \beta$ , come ad esempio parte dei suoi bits, o l'appartenenza o meno ad un dato intervallo. A seconda delle applicazioni crittografiche che si intende realizzare a partire dalla (presunta) funzione one-way dell'esponenziazione modulare, può accadere che la segretezza di tali informazioni parziali risulti vitale per l'integrità dello schema.

Di conseguenza è importante chiarire cosa è possibile dedurre sul valore del  $\text{Dlog } \beta$ , e cosa invece non può essere ricavato, a meno di calcolare per intero il logaritmo discreto.

In letteratura, il tipo di informazione parziale di cui viene esaminata la sicurezza è tipicamente il singolo bit o una sequenza di bits, per cui tale tipo di analisi di sicurezza va sotto il nome di *Bit Security*. A titolo di esempio, si considerino i seguenti problemi riguardo al  $\text{Dlog } \beta$ , in ordine decrescente di difficoltà relativa:

- Noto  $\beta$ , calcolare  $\text{Dlog } \beta$ ;
- Noto  $\beta$ , calcolare una sequenza di bits di  $\text{Dlog } \beta$
- Noto  $\beta$ , calcolare un singolo bit di  $\text{Dlog } \beta$ ;
- Noti  $\beta$  e  $k$  bits del suo logaritmo discreto, calcolare un ulteriore bit di  $\text{Dlog } \beta$ .

Chiaramente la situazione che garantisce maggiore sicurezza è quella in cui anche l'ultimo dei problemi appena accennati risulta tanto difficile quanto calcolare per intero il  $\text{Dlog } \beta$ , nel senso che, supponendo di avere a disposizione un algoritmo efficiente (detto *Oracolo*) per risolvere il problema più semplice, è possibile adoperarlo opportunamente per calcolare, in tempo polinomiale, il logaritmo discreto di  $\beta$ , contro l'assunzione  $\text{Dlog}$ .<sup>9</sup>

---

<sup>9</sup>Si tratta di un classico esempio di *riduzione* di un problema computazionale ad un altro, tecnica utilizzata in *Teoria della Complessità* per stabilire una "gerarchia" nell'ambito dei problemi intrattabili.

Tali circostanze si possono formalizzare mediante le seguenti definizioni:<sup>10</sup>

**Definizione 2.1 (hardcore bit).** Data una funzione one-way  $f : X \rightarrow Y$ , il bit  $i^{\text{esimo}}$  si dice *hardcore* per la funzione  $f$  se ogni algoritmo efficiente per ricavare l' $i^{\text{esimo}}$  bit di  $x$  a partire da  $f(x)$  può essere trasformato in un algoritmo efficiente per invertire la funzione  $f$ .

**Definizione 2.2 ( $k$ -simultaneous hardcore bits).** Data una funzione one-way  $f : X \rightarrow Y$ , i bits alle posizioni  $i_1, \dots, i_k$  si dicono *simultaneamente hardcore* per la funzione  $f$  se, qualunque sia  $1 \leq j \leq k$ , ogni algoritmo efficiente per ricavare lo  $i_j^{\text{esimo}}$  bit di  $x$  a partire dal valore di  $f(x)$  e dalla conoscenza dei bits di  $x$  alle posizioni  $i_1, \dots, i_{j-1}, i_{j+1}, \dots, i_k$  può essere trasformato in un algoritmo efficiente per invertire la funzione  $f$ .

**Definizione 2.3 ( $k$ -sicurezza).** Data una funzione one-way  $f : X \rightarrow Y$ , i bits alle posizioni  $i_1, \dots, i_k$  si dicono *sicuri* se ogni algoritmo efficiente per ricavare tali  $k$  bits di  $x$  a partire dal valore di  $f(x)$ , può essere trasformato in un algoritmo efficiente per invertire la funzione  $f$ .

---

<sup>10</sup>È possibile introdurre delle definizioni più generali, considerando un qualunque predicato su  $x$  anziché semplicemente un suo bit.

Si osservi che la definizione di *k-simultaneous hardcore* bits è quella più stringente, poiché richiede che i bits in questione siano talmente difficile da calcolare che neanche conoscendo  $k - 1$  di essi è possibile ricavare il bit rimanente, senza ricostruire per intero il valore di  $x$ .

Viceversa, la *k-sicurezza*, benché possa apparire simile alla nozione di sicurezza sopra discussa, è in realtà molto più “debole”, in quanto afferma soltanto che ricavare *tutti* i  $k$  bits in esame equivale a determinare  $x$ , mentre potrebbe benissimo darsi il caso che calcolare tutti tranne uno dei  $k$  bits sia facile. In sostanza il problema della sicurezza delle informazioni parziali viene spostato dall'intero valore  $x$  ad una sua porzione più o meno estesa. Tuttavia tale proprietà di sicurezza è comunque utile, poiché consente di puntualizzare meglio le garanzie di intrattabilità che derivano da una data assunzione computazionale.

Da quanto detto segue che nell'esaminare la sicurezza di una funzione one-way in termini della Definizione 2.3, bisogna provare la *k-sicurezza* per  $k$  quanto più piccolo possibile. Si osservi che il caso limite  $k = 1$  coincide con la nozione di *hardcore bit*, che pertanto comporta un livello di sicurezza intermedio rispetto alle Definizioni 2.2 e 2.3.

### 2.3.2 Bit-Security del Logaritmo Discreto

Per chiarire meglio come le definizioni introdotte in precedenza vengano applicate nell'analisi di sicurezza di una funzione one-way, si esamina adesso la *Bit-Security* del Logaritmo Discreto.

Si proverà anzitutto che in  $\mathbb{Z}_p^*$  il bit meno significativo (*Least Significant Bit* o LSB) di  $\text{Dlog } \beta$  è facilmente individuabile, in virtù di alcune proprietà dei gruppi ciclici. D'altra parte, quando  $p \equiv 3 \pmod{4}$ , ogni altro bit di  $\text{Dlog } \beta$  risulta essere *hardcore* nel senso della Definizione 2.1.

#### Come calcolare il LSB del logaritmo discreto di $\beta$

Nel gruppo ciclico  $\mathbb{Z}_p^*$ , si definisce *ordine* di un elemento  $\gamma$  il più piccolo esponente  $c$  per cui si ha che  $\gamma^c \equiv 1 \pmod{p}$ . Inoltre, dalla nozione di elemento *generatore*, risulta che l'ordine di un generatore è pari alla cardinalità del gruppo.

Un'altra nozione utile della *Teoria dei Numeri* è quella di *Quadrato Residuo*, che corrisponde al concetto di quadrato perfetto dell'Aritmetica. Un elemento  $\gamma \in \mathbb{Z}_p^*$  si dice quadrato residuo modulo  $p$  se esiste un  $\delta \in \mathbb{Z}_p^*$  tale che  $\gamma = \delta^2 \pmod{p}$ .

Stabilire se un numero  $\gamma \in \mathbb{Z}_p^*$  è o meno un quadrato residuo risulta possibile in tempo polinomiale, grazie alla seguente proposizione:

**Proposizione 2.1 (Criterio di Eulero).** *Un elemento  $\gamma \in \mathbb{Z}_p^*$  è un quadrato residuo modulo  $p$  se e solo se  $\gamma^{(p-1)/2} \equiv 1 \pmod{p}$ .*

*Dimostrazione.*

( $\Rightarrow$ ) Sia  $\gamma^{(p-1)/2} \equiv 1 \pmod{p}$ . Si consideri un elemento  $\alpha$  generatore del gruppo  $\mathbb{Z}_p^*$ . Allora  $\gamma$  si può scrivere in termini di  $\alpha$ :

$$\exists c \in \mathbb{Z}_{p-1} : \gamma \equiv \alpha^c \pmod{p}$$

e sostituendo nell'ipotesi:

$$\alpha^{(p-1)c/2} \equiv 1 \pmod{p}$$

Essendo  $\alpha$  un generatore, il suo ordine è  $p - 1$ , per cui l'esponente nell'equazione precedente deve essere un multiplo di  $p - 1$ , cioè  $c$  è pari:

$$\exists d \in \mathbb{Z}_{p-1} : c = 2d$$

Pertanto:

$$\begin{aligned}\gamma &\equiv \alpha^c && (\text{mod } p) \\ &\equiv \alpha^{2d} && (\text{mod } p) \\ &\equiv (\alpha^d)^2 && (\text{mod } p) \\ &\equiv \delta^2 && (\text{mod } p)\end{aligned}$$

pur di porre  $\delta = \alpha^d \pmod{p}$ , e quindi  $\gamma$  è un quadrato residuo.

( $\Leftarrow$ ) Si supponga che  $\gamma$  sia un quadrato residuo, ovvero esiste un  $\delta$  di cui  $\gamma$  risulta essere il quadrato (modulo  $p$ ). Per il *Teorema di Eulero*:<sup>11</sup>

$$\delta^{p-1} \equiv 1 \quad (\text{mod } p)$$

Ma allora:

$$\begin{aligned}\gamma^{(p-1)/2} &\equiv (\delta^2)^{(p-1)/2} && (\text{mod } p) \\ &\equiv \delta^{p-1} && (\text{mod } p) \\ &\equiv 1 && (\text{mod } p)\end{aligned}$$

cioè la tesi. □

---

<sup>11</sup>Il *Teorema di Eulero* è un'applicazione al gruppo  $\mathbb{Z}_p^*$  del *Teorema di Lagrange* sull'ordine degli elementi di un gruppo ciclico.

Si noti che se  $\gamma = \delta^2 \pmod p$ , allora  $\gamma = (-\delta)^2 \pmod p$ ; inoltre, dall'essere  $p$  dispari (in quanto  $p$  è un primo grande), si ha che  $\delta$  e  $-\delta \equiv (p - \delta) \pmod p$  sono l'uno pari e l'altro dispari, o viceversa. Pertanto ogni quadrato residuo si ottiene in corrispondenza di due elementi distinti di  $\mathbb{Z}_p^*$ , e quindi il numero di quadrati residui modulo  $p$  è  $(p - 1)/2$ .

Se  $\alpha$  è un generatore di  $\mathbb{Z}_p^*$ , allora tutti gli elementi del tipo  $\alpha^{2c} \pmod p$ , per  $c = 0, 1, \dots, (p - 3)/2$  sono dei distinti quadrati residui; essi sono in numero di  $(p - 1)/2$ , per cui rappresentano tutti i possibili quadrati residui. Di conseguenza un elemento  $\beta$  è un quadrato residuo se e solo se il suo logaritmo discreto è un numero pari, ovvero se il LSB di  $\text{Dlog } \beta$  è 0. D'altra parte, per il *Criterio di Euclide*,  $\beta$  è un quadrato residuo se e solo se  $\beta^{(p-1)/2} \equiv 1 \pmod p$ ; in definitiva, il bit meno significativo di  $\text{Dlog } \beta$  è 0 se e solo se  $\beta^{(p-1)/2} \equiv 1 \pmod p$  (il che è verificabile in tempo polinomiale).

### ***Hardcoreness* dei rimanenti bit del $\text{Dlog } \beta$**

Si discute adesso la sicurezza del bit alla posizione 1 di  $\text{Dlog } \beta$  (cioè il bit immediatamente alla sinistra del LSB), quando  $(p - 1)/2$  è dispari, ovvero  $p \equiv 3 \pmod 4$ . Considerazioni analoghe valgono anche per gli altri bits.

Sia  $\beta$  un quadrato residuo; da quanto discusso in precedenza segue che  $\beta^{(p-1)/2} \equiv 1 \pmod{p}$ , e che  $\beta$  è del tipo  $\alpha^{2c} \pmod{p}$ , con  $c = (\text{Dlog } \beta)/2$ . Se fosse possibile determinare il  $\text{Dlog } \beta$ , si potrebbero ricavare immediatamente le radici quadrate di  $\beta$  modulo  $p$ , calcolando  $\alpha^c \pmod{p}$  (detta *radice quadrata principale di  $\beta$* ), e  $-\alpha^c \pmod{p}$  (*radice quadrata secondaria*).

Nell'ipotesi che  $p \equiv 3 \pmod{4}$ , risulta comunque possibile determinare  $\sqrt{\beta}$ , prescindendo dal calcolo dei logaritmi discreti. Per l'ipotesi, si ha che  $(p+1)/4$  è intero, e risulta che  $\sqrt{\beta} = \pm\beta^{(p+1)/4} \pmod{p}$ . Infatti:

$$\begin{aligned} (\pm\beta^{(p+1)/4})^2 &\equiv \beta^{(p+1)/2} && \pmod{p} \\ &\equiv \beta \cdot \beta^{(p-1)/2} && \pmod{p} \\ &\equiv \beta \cdot 1 && \pmod{p} \\ &\equiv \beta && \pmod{p} \end{aligned}$$

È dunque noto un metodo per estrarre le radici modulari; tuttavia il metodo descritto determina i due valori, ma non consente di discriminare fra la radice principale e quella secondaria.

È interessante notare che riuscire a distinguere fra le due radici consentirebbe di risalire al valore del  $\text{Dlog } \beta$ ; supposto di conoscere il valore di  $\alpha^c \pmod{p}$ , si potrebbe calcolare agevolmente il bit meno significativo di  $c$ , e dall'essere  $c = (\text{Dlog } \beta)/2$ , si ricaverebbe il secondo bit di  $\text{Dlog } \beta$ . A questo punto, se

$\alpha^c \pmod p$  fosse anch'esso un quadrato residuo, sarebbe possibile estrarne la radice quadrata, prendere quella principale, e ricavare, dal LSB di quest'ultimo valore, anche il terzo bit del  $\text{Dlog } \beta$ ; altrimenti, basterebbe considerare  $\alpha^{-1}\alpha^c \equiv \alpha^{(c-1)} \pmod p$ , e ripetere lo stesso ragionamento su tale quadrato residuo. Si ricaverebbe così il logaritmo di  $\beta$  un bit alla volta, ma la procedura terminerebbe comunque in tempo polinomiale.

Per provare che il secondo bit (ovvero quello alla posizione 1) è un *hardcore bit* del logaritmo discreto, basta provare che un algoritmo  $\mathcal{A}$  che consenta di calcolare tale bit a partire da  $\beta$  consentirebbe di discriminare la radice principale di  $\beta$  da quella secondaria.

Si osservi a tal fine che, essendo  $\alpha$  un generatore di  $\mathbb{Z}_p^*$ , esso ha ordine  $p-1$ , per cui  $\alpha^{(p-1)/2} \pmod p$  non può valere 1; l'unico valore possibile è allora  $-1$ , in quanto quadrando bisogna ottenere 1. Quindi  $-\alpha^c \pmod p$  può essere riscritto come:

$$\begin{aligned} -\alpha^c &= -1 \cdot \alpha^c \\ &\equiv \alpha^{(p-1)/2} \alpha^c \pmod p \\ &\equiv \alpha^{c+(p-1)/2} \pmod p. \end{aligned}$$

Ma allora gli esponenti delle due radici quadrate di  $\beta$ , cioè i valori  $c$  e  $c + (p - 1)/2$ , hanno parità diversa, per cui calcolandone il bit meno significativo, mediante il *Criterio di Euclide*, si otterranno sempre valori diversi. D'altra parte, l'oracolo  $\mathcal{A}$  consente di conoscere il secondo bit di  $\text{Dlog } \beta$ , che corrisponde al LSB di  $c$ : dal confronto del valore di tale bit con quelli calcolati con il *Criterio di Euclide*, si riesce a determinare quale fra le due radici,  $\beta^{(p+1)/4} \pmod p$  e  $-\beta^{(p+1)/4} \pmod p$ , è la radice principale di  $\beta$ .

In definitiva, calcolare il secondo bit del  $\text{Dlog } \beta$  consente di discriminare fra la radice principale e quella secondaria di  $\beta$ , e ciò, come già visto, permette di ricavare per intero (in tempo polinomiale) il logaritmo discreto di  $\beta$ . A norma della Definizione 2.1, tale bit è quindi un *hardcore bit* del logaritmo discreto.

Similmente, ogni algoritmo per ricavare l' $i^{\text{esimo}}$  bit del  $\text{Dlog } \beta$ , con  $i \geq 1$ , può essere adoperato come oracolo per una procedura che consente di determinare la radice quadrata principale di  $\beta$ , e pertanto per ogni  $i \in [1, |\beta| - 1]$ , il bit  $i^{\text{esimo}}$  risulta essere *hardcore* per il  $\text{Dlog}$ .

## Capitolo 3

# La funzione Diffie-Hellman

*La funzione **Diffie-Hellman** è una funzione dalle interessanti proprietà crittografiche, la cui difficoltà computazionale è strettamente connessa al problema dei Logaritmo Discreto. Dopo aver evidenziato i legami fra i due problemi, si esaminano le assunzioni computazionali possibili riguardo alla funzione Diffie-Hellman, mostrando alcune questioni di sicurezza fondate su tali ipotesi. Infine, vengono esposti i punti principali riguardo allo stato delle conoscenze sulla sicurezza crittografica della funzione Diffie-Hellman.*

### 3.1 Il Problema Diffie-Hellman

Il problema Diffie-Hellman è il problema la cui difficoltà computazionale costituisce il fondamento della sicurezza del primo schema crittografico ispirato ai principi alla base della Crittografia a Chiave Pubblica.

Come già accennato nella Sezione 1.3, in [DH76] Diffie ed Hellman proposero uno schema per risolvere la questione dello scambio della chiave segreta, necessario per garantire la *confidenzialità* di una susseguente comunicazione fra due parti, Alice e Bob, che facciano uso di un cifrario convenzionale. Tale schema, noto come *Schema Diffie-Hellman per lo scambio di chiavi*, può essere descritto come segue:

- Alice e Bob scelgono pubblicamente un numero primo  $p$  per cui il problema del logaritmo discreto in  $\mathbb{Z}_p^*$  sia (probabilmente) intrattabile, ed un elemento  $\alpha$  generatore di  $\mathbb{Z}_p^*$ ;
- Alice sceglie e mantiene segreto un numero  $a \in \mathbb{Z}_{p-1}$ , calcola il valore  $\beta = \alpha^a \pmod p$ , e spedisce  $\beta$  a Bob;
- Bob sceglie e mantiene segreto un numero  $b \in \mathbb{Z}_{p-1}$ , calcola il valore  $\gamma = \alpha^b \pmod p$ , e spedisce  $\gamma$  ad Alice;

- Bob, quando riceve  $\beta$ , calcola  $k_B = \beta^b \pmod p$ , e fissa  $k = k_B$  quale chiave per la susseguente conversazione con Alice;
- Alice, quando riceve  $\gamma$ , calcola  $k_A = \gamma^a \pmod p$ , e fissa  $k = k_A$  quale chiave per la susseguente conversazione con Bob.

Al termine di questo protocollo,<sup>1</sup> Alice e Bob condividono una chiave  $k$  alla cui determinazione hanno contribuito entrambi, e che solo loro conoscono. Si osservi innanzitutto che il valore  $k$  calcolato dalle due parti è lo stesso:

$$\begin{aligned}
 k_A &\equiv \gamma^a \pmod p \\
 &\equiv (\alpha^b)^a \pmod p \\
 &\equiv (\alpha^a)^b \pmod p \\
 &\equiv \beta^b \pmod p \\
 &\equiv k_B \pmod p
 \end{aligned}$$

Resta da chiarire perché solo Alice e Bob siano in grado di determinare  $k$ .

Si consideri un avversario Oscar, che ha intercettato tutta la comunicazione.

Oscar conosce i valori  $p, \alpha, \beta = \alpha^a \pmod p$ , e  $\gamma = \alpha^b \pmod p$ , e vuole ricavare

---

<sup>1</sup>In effetti, si tratta di una versione semplificata dello schema Diffie-Hellman per lo scambio delle chiavi, che risulta passibile di attacchi da parte di un avversario passivo (*person-in-the-middle attack*) e da parte di un avversario attivo (*reply attack*). Entrambi questi attacchi possono essere evitati se si combina lo scambio delle chiavi con uno schema di firma digitale, che consente alle due parti di autenticare mutualmente le loro identità.

$k = \text{DH}(\alpha^a \bmod p, \alpha^b \bmod p) = \alpha^{ab} \bmod p$ . La funzione DH introdotta è proprio la funzione Diffie-Hellman, ed il problema che Oscar deve risolvere è detto *Problema Diffie-Hellman* (DHP):

INPUT:  $p$ ,  $\alpha$  generatore di  $\mathbb{Z}_p^*$ ,  $\alpha^a \bmod p$  e  $\alpha^b \bmod p$

OUTPUT:  $\alpha^{ab} \bmod p$

Se Oscar fosse in grado di calcolare i logaritmi discreti in  $\mathbb{Z}_p^*$ , potrebbe agevolmente calcolare  $\alpha^{ab} \bmod p$  (detto anche *segreto* Diffie-Hellman), ad esempio calcolando  $a = \text{Dlog } \beta$  e quindi  $k = \gamma^a \bmod p$ . Si potrebbe quindi essere tentati di dire che la sicurezza del problema Diffie-Hellman si basa sull'assunzione Dlog, ma ciò sarebbe errato, in quanto l'osservazione precedente dimostra soltanto che esiste una riduzione polinomiale da DHP a Dlog, e non che i due problemi sono (polinomialmente) equivalenti. In definitiva, è possibile affermare che il DHP è *al più* tanto difficile quanto il Dlog, ma l'equivalenza fra i due problemi costituisce un'importante questione irrisolta in ambito crittografico.

Recentemente, parecchio lavoro è stato dedicato dalla comunità scientifica a tale interrogativo, nel tentativo di risolvere la questione in un senso o nell'altro. Alcuni fra i risultati più interessanti sono stati provati in [Mau94]. In tale articolo si prova che, quando il gruppo  $\mathcal{G}$  in cui si opera soddisfa particolari

condizioni, risolvere il problema Diffie-Hellman risulta essere computazionalmente equivalente a calcolare i logaritmi discreti, nel senso che ogni algoritmo per determinare  $\alpha^{ab} \bmod p$  a partire da  $\alpha^a \bmod p$  e  $\alpha^b \bmod p$ , può essere utilizzato come oracolo per costruire una procedura che, noto  $\beta$ , calcoli  $\text{Dlog } \beta$ .

Per formulare tali condizioni si fa uso della nozione di *B-smoothness*, introdotta nella seguente definizione:

**Definizione 3.1 (B-smoothness).** Un numero  $m$  si dice *B-smooth* (*liscio rispetto alla soglia  $B \geq 0$* ) se tutti i suoi fattori primi sono maggiorati da  $B$ .

Sussistono, allora, i seguenti risultati [Mau94].

- Sia  $p$  un numero primo, e si supponga che la fattorizzazione di  $p - 1$  sia nota. Sia inoltre  $B$  una soglia polinomiale rispetto alla dimensione di  $p$ , cioè  $B = \mathcal{O}((\log p)^c)$ , per qualche costante  $c$ . Se  $\varphi(p - 1)$  è *B-smooth*, allora il problema Diffie-Hellman e il problema del logaritmo discreto sono computazionalmente equivalenti nel gruppo in  $\mathbb{Z}_p^*$ .
- Più in generale, sia  $\mathcal{G}$  un gruppo ciclico finito di ordine  $n$ , di cui sia nota la fattorizzazione. Sia inoltre  $B$  una soglia tale che  $B = \mathcal{O}((\log n)^c)$ , per qualche costante  $c$ . Se  $\varphi(n)$  è *B-smooth*, allora in  $\mathcal{G}$  il problema Diffie-Hellman e il problema del logaritmo discreto sono computazionalmente equivalenti nel gruppo  $\mathcal{G}$ .

## 3.2 Assunzioni Computazionali legate al DHP

Si consideri di nuovo lo schema Diffie-Hellman per lo scambio di chiavi. Per quanto detto in precedenza, non è possibile basare, in generale, la sicurezza dello schema su alcuna ipotesi computazionale nota; è pertanto opportuno introdurre una nuova assunzione, detta *Assunzione Computazionale Diffie-Hellman* o CDH, che avanza la congettura che calcolare il segreto dello scambio Diffie-Hellman sia un problema computazionale intrattabile.

L'importanza di tale assunzione deriva dal fatto che anche altri schemi crittografici, la cui sicurezza sembrerebbe riconducibile al problema Dlog, si fondano in effetti sull'ipotesi CDH. Si consideri ad esempio il Crittosistema ElGamal: si tratta del primo schema di codifica a chiave pubblica randomizzato, nel senso che il cifrato corrispondente ad un dato messaggio in chiaro non è unico, e tuttavia il destinatario è in grado di ricavare il testo in chiaro originale.

### Crittositema ElGamal

Si ricordi che in uno schema di criptazione a chiave pubblica, ogni partecipante deve generare la propria coppia  $\langle \textit{chiave pubblica}, \textit{chiave privata} \rangle$ .

Per generare tale coppia, si sceglie in maniera casuale un primo  $p$  per cui risulti che tutti gli algoritmi noti per il calcolo del logaritmo discreto in  $\mathbb{Z}_p^*$  richiedano tempo super-polinomiale. Si sceglie inoltre un generatore  $\alpha$  di  $\mathbb{Z}_p^*$ , e un  $a \in \mathbb{Z}_{p-1}$ , che viene tenuto segreto. Si calcola  $\beta = \alpha^a \pmod p$ , e si pone:

- *chiave pubblica*  $\doteq (p, \alpha, \beta)$ ;
- *chiave privata*  $\doteq a$

Per cifrare un testo in chiaro  $x \in \mathbb{Z}_p^*$ , si sceglie in maniera casuale un numero  $r \in \mathbb{Z}_{p-1}$ , (questa è l'operazione che introduce un certo grado di non-determinismo nel processo), e si utilizza la chiave pubblica calcolando:

$$(y_1, y_2) = (\alpha^r \pmod p, x\beta^r \pmod p)$$

Viceversa, per decifrare un crittogramma  $(y_1, y_2)$ , il proprietario della chiave segreta calcola:

$$x' = y_2(y_1^a)^{-1} \pmod p$$

Si noti che  $x' = x$  poiché:

$$\begin{aligned}
 x' &\equiv y_2(y_1^a)^{-1} &&\equiv y_2((\alpha^r)^a)^{-1} &&(\text{mod } p) \\
 &\equiv y_2((\alpha^a)^r)^{-1} &&\equiv y_2(\beta^r)^{-1} &&(\text{mod } p) \\
 &\equiv x\beta^r(\beta^r)^{-1} &&\equiv x \cdot 1 &&(\text{mod } p) \\
 &\equiv x &&&&(\text{mod } p)
 \end{aligned}$$

Per rompere il sistema, un avversario deve riuscire a risalire a  $x$ , conoscendo  $p, \alpha, \beta, y_1, y_2$ . Per provare l'equivalenza di tale compito con il problema Diffie-Hellman, si mostra come un algoritmo per risolvere l'uno possa essere adoperato come oracolo da una procedura che risolve l'altro.

Si supponga di avere a disposizione un algoritmo  $\mathcal{A}$  per risolvere DHP, ovvero:

$$\mathcal{A}(p, \alpha, \alpha^a \pmod p, \alpha^b \pmod p) = \alpha^{ab} \pmod p.$$

Per rompere il crittosistema ElGamal, si consideri:

$$\begin{aligned}
 \mathcal{A}(p, \alpha, y_1, \beta) &= \mathcal{A}(p, \alpha, \alpha^r \pmod p, \alpha^a \pmod p) \\
 &= \alpha^{ra} \pmod p \\
 &= \beta^r \pmod p
 \end{aligned}$$

Pertanto, per risalire ad  $x$ , si può calcolare:

$$\begin{aligned} y_2(\beta^r)^{-1} &\equiv x\beta^r(\beta^r)^{-1} \pmod{p} \\ &\equiv x \pmod{p} \end{aligned}$$

Viceversa, si supponga di avere a disposizione un algoritmo  $\mathcal{B}$  per rompere ElGamal, ovvero:

$$\mathcal{B}(p, \alpha, \beta, y_1, y_2) = x$$

Si osservi che, in questo caso, il segreto del problema Diffie-Hellman risulta essere  $\alpha^{ar} \pmod{p}$ ; per calcolarlo, si interroghi l'oracolo  $\mathcal{B}$  sul crittogramma  $(y_1, y_2) = (\alpha^r \pmod{p}, 1)$ :

$$\begin{aligned} \mathcal{B}(p, \alpha, \beta, \alpha^r \pmod{p}, 1) &= \mathcal{B}(p, \alpha, \beta, \alpha^r \pmod{p}, ((\beta^r)^{-1} \pmod{p}) \cdot \beta^r) \\ &= (\beta^r)^{-1} \pmod{p} \\ &= ((\alpha^a)^r)^{-1} \pmod{p} \\ &= (\alpha^{ar})^{-1} \pmod{p} \end{aligned}$$

Per ottenere  $\alpha^{ar} \pmod{p}$  basta invertire tale valore, per cui resta provato che la sicurezza del crittosistema ElGamal coincide con l'assunzione CDH.

In effetti, la nozione di sicurezza sopra esaminata non è molto robusta, poiché esclude soltanto che un avversario riesca a comprendere *per intero* i messaggi cifrati, ma potrebbe comunque riuscire a capire di che tipo di messaggio

si tratti.

Una nozione di sicurezza più stringente è quella di *sicurezza semantica*:

**Definizione 3.2 (Semantic security).** Un crittosistema  $\mathcal{C}$  è *semanticamente sicuro* se ogni avversario polinomiale, scelto opportunamente un messaggio  $\mathcal{M}$ , e ricevuto indietro un crittogramma  $\mathcal{M}'$  che può essere o la criptazione di  $\mathcal{M}$  o la criptazione di un messaggio completamente random  $\mathcal{R}_{\mathcal{M}}$ , non riesce a distinguere fra le due circostanze.

In sostanza la cifratura non lascia trasparire alcunché sul significato del messaggio, in quanto non si riesce a distinguere fra il crittogramma corrispondente ad un testo di propria scelta (e quindi possibilmente con un alto livello di ridondanza), e il crittogramma relativo ad un messaggio completamente privo di struttura semantica.

A questo punto ci si potrebbe domandare: il crittosistema ElGamal soddisfa la definizione di sicurezza semantica? Ancora una volta, è necessario fare delle ipotesi opportune per riuscire a dare una risposta (affermativa) al quesito posto. Purtroppo, al momento non è noto se l'assunzione CDH consenta o meno di provare la sicurezza semantica di ElGamal. È tuttavia possibile introdurre una nuova ipotesi, anch'essa legata al DHP, la cui veridicità equivale

a rispondere positivamente alla domanda in esame.

L'assunzione di cui si parla è nota come *Decisional Diffie-Hellman* o DDH, e (come si intuisce dal nome) riguarda un problema *decisionale* piuttosto che *computazionale*. Tale ipotesi sostiene che, fissato un primo  $p$  per cui il problema del logaritmo discreto sia difficile, non è possibile distinguere (in tempo polinomiale) fra una quaterna del tipo:

$$(\alpha, \alpha^a \pmod p, \alpha^b \pmod p, \alpha^{ab} \pmod p) \quad \text{tipo } \textit{Diffie-Hellman}$$

ed una quaterna del tipo:

$$(\alpha, \alpha^a \pmod p, \alpha^b \pmod p, \alpha^c \pmod p) \quad \text{tipo Random}$$

Per provare che la sicurezza semantica del crittosistema ElGamal equivale a distinguere quaterne del tipo “Diffie-Hellman” da quaterne del tipo “Random” (ovvero a *decidere Diffie-Hellman*), si procede ad una riduzione simile a quanto visto in precedenza dimostrando l'equivalenza della sicurezza di ElGamal con il problema CDH.

Si supponga di avere a disposizione un algoritmo  $\mathcal{D}$  per decidere DHP:

$$\mathcal{D}(\alpha, \alpha^a \pmod p, \alpha^b \pmod p, \delta) = \begin{cases} 1 & \text{se } \delta = \alpha^{ab} \pmod p \\ 0 & \text{altrimenti} \end{cases}$$

Si consideri adesso un avversario Oscar che sceglie un messaggio  $x$  e ottiene un suo “presunto” crittogramma  $(y_1, y_2)$ . Per capire se si tratta veramente di una criptazione di  $x$ , Oscar calcola  $\mathcal{D}(\alpha, \beta, y_1, (x^{-1} \bmod p) \cdot y_2)$ : se ottiene 1, allora gli è stata fornita una criptazione genuina di  $x$ ; altrimenti, si tratta della criptazione di un testo casuale. Se infatti  $(y_1, y_2)$  è una crittogramma di  $x$ , risulta

$$\begin{aligned} \mathcal{D}(\alpha, \beta, y_1, (x^{-1} \bmod p) \cdot y_2) &= \mathcal{D}(\alpha, \alpha^a \bmod p, \alpha^r \bmod p, (x^{-1} \bmod p) \cdot (x\beta^r)) \\ &= \mathcal{D}(\alpha, \alpha^a \bmod p, \alpha^r \bmod p, 1 \cdot \alpha^{ar}) \\ &= 1 \end{aligned}$$

Se invece  $(y_1, y_2)$  non è un crittogramma valido per  $x$ , l’oracolo  $\mathcal{D}$  restituirà 0.

Viceversa, si supponga di avere a disposizione un oracolo  $\mathcal{O}$  in grado di rompere la sicurezza semantica di ElGamal, ovvero:

$$\mathcal{O}(\alpha, \beta, y_1, y_2, x) = \begin{cases} 1 & \text{se } x = y_2(y_1^a)^{-1} \bmod p \\ 0 & \text{altrimenti} \end{cases}$$

Oscar viene messo di fronte alla quaterna  $(\alpha, \alpha^a \bmod p, \alpha^b \bmod p, \alpha^c \bmod p)$  e deve decidere di che tipo essa sia, cioè deve stabilire se:

$$\alpha^c \equiv \alpha^{ab} \pmod{p}$$

A tal fine, Oscar chiede all'oracolo  $\mathcal{O}$  se  $(\alpha^b \bmod p, \alpha^c \bmod p)$  è un crittogramma legittimo per  $x = 1$ : se ottiene 1, allora la quaterna in esame è di tipo Diffie-Hellman, altrimenti è di tipo Random. Infatti:

$$\mathcal{O}(\alpha, \alpha^a \bmod p, \alpha^b \bmod p, \alpha^c \bmod p, 1) = 1$$

$$\Updownarrow$$

$$1 \equiv \alpha^c \cdot [(\alpha^b)^a]^{-1} \equiv \alpha^c \cdot (\alpha^{ab})^{-1} \pmod{p}$$

$$\Updownarrow$$

$$\alpha^c \equiv \alpha^{ab} \pmod{p}$$

Ciò prova l'equivalenza della sicurezza semantica del Crittosistema ElGamal con l'assunzione DDH.

Si osservi infine che è possibile introdurre un'ipotesi decisionale (come l'assunzione DDH) riguardo alla funzione Diffie-Hellman soltanto in virtù del fatto che non si tratta di una funzione one-way vera e propria. Tale funzione risulta infatti difficile da computare in entrambe le direzioni.

### 3.3 Sicurezza della funzione Diffie-Hellman

Come già osservato, l'assunzione CDH implica l'assunzione Dlog, il che viene solitamente denotato scrivendo:  $\text{CDH} \leq_P \text{Dlog}$ . D'altra parte, risulta ovviamente che  $\text{DDH} \leq_P \text{CDH}$ , in quanto se fosse semplice scoprire il segreto Diffie-Hellman, sarebbe possibile distinguere efficientemente quaterne del tipo Diffie-Hellman da quaterne del tipo Random. Unendo le due relazioni:

$$\text{DDH} \leq_P \text{CDH} \leq_P \text{Dlog}$$

Attualmente non è noto se una (o entrambe) delle relazioni riportate sopra sia stretta. Ciò che è possibile dire è che l'unico modo noto per *decidere* Diffie-Hellman consiste nel *calcolarne* il segreto, il che a sua volta deve essere ricondotto al calcolo di un Logaritmo Discreto. Tuttavia, è arduo credere che si tratti del modo migliore di procedere: ciò significherebbe, in altri termini, che *verificare* la relazione  $\alpha^c \equiv \alpha^{ab} \pmod{p}$  richiede lo stesso sforzo computazionale di *effettuare per intero i relativi calcoli*.

Un avanzamento verso una migliore comprensione della questione si è avuto di recente grazie al lavoro di alcuni ricercatori, che hanno costruito dei modelli matematici nell'ambito dei quali studiare in dettaglio l'assunzione DDH. Al riguardo, si analizzano brevemente i risultati presentati in [Sho97].

Nel tentativo di provare che l'assunzione DDH sia erronea, si potrebbe cercare di realizzare un algoritmo generico in grado di decidere efficientemente il problema Diffie-Hellman in un qualunque gruppo  $\mathcal{G}$ . Un algoritmo generico di tale tipo è stato già esposto discutendo del problema del logaritmo discreto: si tratta dell'algoritmo di Shanks, che però risulta troppo lento, avendo un tempo di esecuzione  $\mathcal{O}(\sqrt{p})$ . Al contrario, il metodo dell'*Index Calculus* non è un algoritmo generico, poiché esso si basa su particolari proprietà valide in  $\mathbb{Z}_p^*$ , ma non applicabili in generale (ad esempio, l'*Index Calculus* non può essere adoperato nel gruppo formato dall'insieme dei punti di una curva ellittica).

In [Sho97], Shoup mostra che non può esistere alcun algoritmo generico che decida Diffie-Hellman in tempo  $\Omega(\sqrt{p})$ . Pertanto, il miglior algoritmo generico per decidere Diffie-Hellman consiste nel calcolare per intero un logaritmo discreto mediante l'algoritmo di Shanks, per poi verificare se la quaterna in esame verifica o meno la relazione  $\alpha^c \equiv \alpha^{ab} \pmod{p}$ .

Quanto detto ha delle interessanti conseguenze. Il risultato non prova che per decidere Diffie-Hellman non si possa fare di meglio che utilizzare l'algoritmo di Shanks: esso piuttosto mette in chiaro che algoritmi più efficienti possono essere ottenuti solo affrontando il problema in un gruppo specifico, e

sfruttando opportunamente le particolari caratteristiche che tale gruppo dovesse presentare (come ad esempio avviene nel metodo dell'*Index Calculus*, che opera in  $\mathbb{Z}_p^*$ ). Potrebbe pertanto darsi il caso che in ogni gruppo  $\mathcal{G}$  sia possibile seguire un approccio speciale che decida efficientemente la questione; ciò che risulta certo è che ogni approccio generale al problema è votato al fallimento.

### 3.3.1 DDH versus CDH

Riassumendo quanto affermato in precedenza, allo stato attuale delle conoscenze l'assunzione DDH presenta l'interessante caratteristica di non soffrire di attacchi polinomiali, e al tempo stesso di consentire di dimostrare le proprietà di sicurezza di altre primitive crittografiche. Si esamina adesso un caso in cui l'assunzione DDH consente di migliorare l'efficienza di una possibile applicazione crittografica, senza comprometterne la sicurezza.

Il protocollo Diffie-Hellman per lo scambio di chiavi, introdotto in precedenza, consente a due parti, Alice e Bob, di condividere una chiave di sessione  $k$  la cui segretezza è garantita dall'assunzione CDH.

Sebbene gli algoritmi noti per il calcolo del logaritmo discreto siano inefficienti, grazie alla capacità di calcolo dell'hardware disponibile oggi è

possibile determinare logaritmi discreti in un tempo non troppo lungo,<sup>2</sup> quando si tratta di gruppi con meno  $10^{130}$  elementi. È pertanto pratica comune utilizzare (nel caso del gruppo  $\mathbb{Z}_p^*$ ) numero primi aventi almeno 300 cifre decimali, ovvero circa 1024 bits. Di conseguenza, la chiave segreta  $k$  ottenuta alla fine del protocollo Diffie-Hellman ha anch'essa circa 1024 bits, essendo un intero ridotto modulo  $p$ .

Tipicamente, però, lo scopo dello scambio di chiavi è quello di ottenere una chiave di sessione da utilizzare con cifrari simmetrici, come ad esempio il DES, che fa uso di chiavi segrete a 56 bits. Si potrebbe allora pensare di ricavare tale chiave a partire da  $k$ , prendendone, per esempio, solamente i 56 bits più significativi. Ciò tuttavia renderebbe insicura la comunicazione, in quanto l'assunzione CDH consente di dimostrare soltanto che un avversario Oscar non possa calcolare per intero  $k$ , ma non garantisce (direttamente) che Oscar non sia in grado di ricavarne una parte, magari proprio i primi 56 bits.

La questione appena introdotta riguarda la *Bit Security*, e deve quindi essere analizzata facendo riferimento alle Definizioni 2.1, 2.2 e 2.3, introdotte nel capitolo precedente.

---

<sup>2</sup>Si noti che, in ambito crittografico, viene ritenuto “non troppo lungo” un tempo dell'ordine di una decina d'anni; ciò per mettersi al riparo da eventuali improvvisi avanzamenti tecnologici e del software, che dovessero portare ad una notevole velocizzazione degli algoritmi adoperati.

Un importante risultato in ambito di *Bit Security* afferma che ogni funzione one-way  $f(x)$  ha almeno un *predicato hardcore* [GL89], ovvero un predicato che è tanto difficile da calcolare quanto lo è invertire la funzione stessa.

Questo risultato si applica anche alla funzione Diffie-Hellman; mentre però per alcune funzioni one-way  $f(x)$  tale predicato è semplicemente uno dei bits di  $x$  (si veda ad esempio la Sezione 2.3.2 per il caso del logaritmo discreto), nel caso della funzione Diffie-Hellman si tratta del bit risultante dal prodotto interno<sup>3</sup> fra  $\text{DH}(\alpha^a \bmod p, \alpha^b \bmod p)$  e una stringa di bits random (ma pubblicamente nota) di pari lunghezza.

Tale predicato è l'unico predicato *hardcore* che si conosca per la funzione Diffie-Hellman, quando si considera l'assunzione CDH. Pertanto, dei 1024 bits della chiave  $k$  ottenuta al termine di un'iterazione del protocollo Diffie-Hellman, è possibile estrarre un solo bit la cui imprevedibilità segua dalle assunzioni fatte. Di conseguenza, per essere certi che tutti i 56 bits della chiave di sessione risultino sicuri, è necessario ripetere 56 volte lo scambio: in tal modo si ottiene senza dubbio un protocollo sicuro e polinomiale, ma non molto efficiente.

---

<sup>3</sup>Si definisce prodotto interno di due stringhe di bits di pari lunghezza  $x$  e  $y$  lo *xor* dell'*and* dei bits alle posizioni omonime. Denotando con  $t$  la lunghezza comune, con  $x_i$  e  $y_i$  il bit alla posizione  $i$  rispettivamente di  $x$  e di  $y$ , si ha:

$$\langle x, y \rangle = \bigoplus_{i=0}^{t-1} x_i y_i$$

D'altra parte, se si basasse l'analisi di sicurezza sull'assunzione DDH, sarebbe possibile ricavare per intero la chiave di sessione a partire da un'unica chiave  $k$  a 1024 bits, eseguendo una sola iterazione del protocollo Diffie-Hellman; ciò perché se Oscar riuscisse a calcolare alcunché riguardo a  $k$  (in particolare qualche parte della chiave di sessione), disporrebbe di una tecnica per distinguere quaterne Diffie-Hellman da quaterne Random, contro l'assunzione DDH.

In definitiva, l'utilizzo dell'assunzione DDH consente di progettare una versione più efficiente del protocollo, senza tuttavia rinunciare ad una rigorosa dimostrazione di sicurezza. L'inconveniente è che l'ipotesi sulla quale ci si basa, per quanto supportata dai risultati parziali esposti in precedenza, è in effetti meno solida dell'assunzione CDH o Dlog.

Si noti infine che, sotto l'assunzione CDH, sebbene non sia possibile provare che esistano 56 bits di  $k$  che risultino singolarmente difficili da calcolare (*hardcore*), è tuttavia possibile dimostrare che i primi 56 bits di  $k$  risultano *sicuri* nel senso della Definizione 2.3: tale risultato, presentato in [BV96], costituisce, insieme ad una sua estensione realizzata nell'ambito del presente lavoro di Tesi, l'oggetto del capitolo seguente.

# Capitolo 4

## Bit Security della funzione

### Diffie-Hellman

*In questo capitolo si analizza una questione di Bit Security riguardante la funzione Diffie-Hellman: la  $k$ -sicurezza del segreto  $DH(\alpha^a \bmod p, \alpha^b \bmod p)$ .*

*Inizialmente, si affronta il caso dei primi  $k$  bits del segreto, e si introduce il Problema del Numero Nascosto, che può essere efficientemente risolto utilizzando delle tecniche di Approssimazione nei Reticoli.*

*Viene quindi presentato un nuovo risultato teorico, elaborato nell'ambito del presente lavoro di Tesi, che consente di generalizzare i risultati ottenuti al caso di una qualunque finestra di bits centrali.*

## 4.1 Il Problema del Numero Nascosto

### 4.1.1 Introduzione

Nella risoluzione di talune questioni di sicurezza, è spesso conveniente ricondurre lo studio all'analisi di un problema più astratto, opportunamente definito. Nel caso della *k*-sicurezza (nel senso della Definizione 2.3) del segreto Diffie-Hellman, è possibile fare riferimento al *Problema del Numero Nascosto*. Nella prossima sezione se ne presenta una possibile formulazione, adottando la notazione introdotta in [BV96]; qui di seguito, invece, si introducono alcune nozioni preliminari.

Gli elementi del gruppo  $\mathbb{Z}_{p-1}$ , con  $p$  primo, possono essere rappresentati in formato binario, utilizzando  $n \doteq \lceil \log p \rceil$  bits, che consentono di descrivere  $2^n > p$  elementi. Si deve quindi gestire questo “eccesso” di potenza espressiva, che potrebbe risultare inconveniente, in quanto introduce un *bias* sul bit più significativo. Tale bit infatti risulterà essere più spesso 0 che non 1, in quanto solo nei  $(p \bmod 2^{n-1})$  elementi di  $\mathbb{Z}_{p-1}$  maggiori di  $2^{n-1}$  si utilizza il MSB.

Trattando questioni di sicurezza che riguardano tale bit, spesso si preferisce adottare un'altra nozione di *MSBs*, introdotta nel campo della Crittografia da Micali e Goldwasser.

**Definizione 4.1 (Micali-Goldwasser MSBs Notation).**

Dato un elemento  $x \in \mathbb{Z}_{p-1}$ , i  $k$  bits più significativi di  $x$   $\text{MSB}_k(x)$  secondo Micali-Goldwasser sono rappresentati dall'intero  $t \in \{0, \dots, 2^k - 1\}$  per cui risulta che  $t \cdot p/2^k \leq x \leq (t + 1) \cdot p/2^k$ .

In altri termini, il gruppo  $\mathbb{Z}_{p-1}$  viene suddiviso in  $2^k$  intervalli di ampiezza pressoché uguale, numerati da 0 a  $2^k - 1$ ; a questo punto, si denota con  $\text{MSB}_k(x)$  l'indice relativo all'intervallo in cui cade l'elemento in questione. Nel seguito però, per convenienza, si preferisce denotare con  $\text{MSB}_k(x)$  l'elemento di  $\mathbb{Z}_{p-1}$  che occupa la posizione mediana dell'intervallo cui  $x$  appartiene; ciò perché in tal modo sussiste la relazione:

$$|x - \text{MSB}_k(x)| < p/2^{k+1}$$

**4.1.2 Possibili formulazioni del Problema**

Vengono nel seguito definite alcune possibili varianti del Problema del Numero Nascosto. Di queste, una è facilmente risolvibile, mentre per affrontare le altre, che risultano equivalenti, è necessario fare ricorso ad alcuni strumenti di *Teoria dei Numeri*, descritti, per completezza, nell'Appendice A.

### Problema del Numero Nascosto (Hidden Number Problem)

Sia assegnato, in  $\mathbb{Z}_p^*$ , un valore  $\gamma$  incognito, detto *numero nascosto*. Il *Problema del Numero Nascosto* consiste nel determinare tale  $\gamma$  in tempo polinomiale in  $(\log p)$ , avendo accesso ad un oracolo  $\mathcal{O}_\gamma(\xi)$  che, per ogni  $\xi \in \mathbb{Z}_p^*$ , è in grado di calcolare i  $k$  bits più significativi (nel senso della Definizione 4.1) dell'elemento ottenuto mascherando  $\gamma$  con  $\xi$ :

$$\mathcal{O}_\gamma(\xi) = \text{MSB}_k(\gamma \cdot \xi \bmod p)$$

La possibilità di interrogare l'oracolo  $\mathcal{O}_\gamma$ , su dei valori “maschera”  $\xi$  a propria scelta, consente di risolvere facilmente il problema: è sufficiente eseguire *queries* del tipo  $\mathcal{O}_\gamma(2^{ki})$ , per opportuni valori di  $i$ , per risalire,  $k$  bits alla volta, all'intero valore di  $\gamma$ . Tale tecnica è utilizzata in [ACGS88], dove gli autori analizzano la *Bit Security* della funzione RSA e della funzione di Rabin.

Si potrebbe allora tentare di affrontare la questione “con gli occhi chiusi”, vale a dire interrogando l'oracolo  $\mathcal{O}_\gamma$  su elementi a caso, cercando poi di estrapolare, dalle risposte ottenute, l'elemento nascosto. Si ottiene così il *Problema Randomizzato del Numero Nascosto (Sampling Hidden Number Problem)*:

INPUT:  $(\xi_1, \mathcal{O}_\gamma(\xi_1)), \dots, (\xi_d, \mathcal{O}_\gamma(\xi_d))$  con  $\xi_i \in \mathbb{Z}_p^*$

OUTPUT:  $\gamma$

La questione interessante è risolvere questo problema minimizzando simultaneamente sia il numero  $k$  di bits restituiti dall'oracolo ad ogni *query*, sia il numero  $d$  di coppie  $(\xi_i, \mathcal{O}_\gamma(\xi_i))$  fornite in input all'algoritmo.

Prima di passare ad esporre un metodo polinomiale per risolvere anche questa versione del problema, si osserva che, sotto l'assunzione Dlog, la versione *sampling* può essere riformulata considerando un nuovo oracolo  $\mathcal{O}_{\gamma, \alpha}$  che risponde soltanto a queries relative ad elementi “maschera”  $\alpha^x \bmod p$ , di cui sia noto il logaritmo discreto  $x$  (rispetto al generatore  $\alpha$ ). Più precisamente:

#### Problema Randomizzato del Numero Nascosto

Sia assegnato, in  $\mathbb{Z}_p^*$ , un generatore  $\alpha$  noto, ed un valore  $\gamma$  incognito, detto *numero nascosto*. Il *Problema Randomizzato del Numero Nascosto* consiste nel determinare il valore  $\gamma$  in tempo polinomiale in  $(\log p)$ , avendo accesso ad un oracolo  $\mathcal{O}_{\gamma, \alpha}(x)$  che, per ogni  $x \in \mathbb{Z}_{p-1}$ , è in grado di calcolare i  $k$  bits più significativi (nel senso della Definizione 4.1) dell'elemento ottenuto mascherando  $\gamma$  con  $\alpha^x \bmod p$ :

$$\mathcal{O}_{\gamma, \alpha}(x) = \text{MSB}_k(\gamma \cdot \alpha^x \bmod p)$$

Il motivo per cui tale formulazione è equivalente a quella in cui si hanno a disposizione solo coppie *random*  $(\xi_i, \mathcal{O}_\gamma(\xi_i))$  è che, non essendo possibile

calcolare in maniera efficiente i logaritmi discreti, l'unico modo per fornire all'oracolo una "maschera" di cui si conosca il Dlog è scegliere un esponente  $x$  e calcolare successivamente  $\alpha^x \bmod p$ , ottenendo così dei valori sui quali non si ha alcun controllo, e che possono pertanto essere considerati come quantità casuali.

Vista l'equivalenza fra i due modi di presentare il Problema Randomizzato del Numero Nascosto, nell'esporre la soluzione si utilizzerà alternativamente l'una o l'altra forma, a seconda di quale risulti più espressiva. Ciò non comporterà confusione, pur di ricordare la relazione  $\xi_i = \alpha^{x_i} \bmod p, 1 \leq i \leq d$ .

### 4.1.3 Risoluzione per Approssimazione in un Reticolo

Per risolvere la versione randomizzata del Problema del Numero Nascosto, si procede riconducendo la questione ad un Problema di Approssimazione nei Reticoli. Gli algoritmi ai quali si farà riferimento nel seguito sono descritti nell'Appendice A.

Si osservi innanzitutto che per scoprire un elemento incognito  $\gamma \in \mathbb{Z}_p^*$ , è necessario disporre di almeno  $n$  bits di informazione, dove  $n \doteq \lceil \log p \rceil$ . Si ricordi che nella versione *sampling* del problema, l'informazione viene fornita

sotto forma di  $d$  coppie, ciascuna contenente i  $k$  MSBs del prodotto di  $\gamma$  con un numero  $\xi$  uniformemente distribuito in  $\mathbb{Z}_p^*$ . Dal momento che si è interessati a minimizzare entrambi i parametri  $d$  e  $k$ , la situazione più equilibrata per cui  $\gamma$  rimane univocamente determinato sarebbe  $k = \lceil \sqrt{n} \rceil$  e  $d = \lceil \sqrt{n} \rceil$ .

Si analizza adesso una soluzione al problema posto, dovuta a Boneh e Venkatesan [BV96], che si avvicina a tale limite minimo, avendo bisogno di  $2\lceil \sqrt{n} \rceil$  responsi di un oracolo che restituisca  $\lceil \sqrt{n} \rceil + \lceil \log n \rceil$  MSBs per query.

Tale soluzione è basata sull'idea di costruire un Reticolo a partire dai valori degli elementi casuali  $\xi_i$  di cui si conosce l'output  $\mathcal{O}_{\gamma, \alpha}(\xi_i)$  restituito dall'oracolo. Procedendo così si impone al Reticolo una particolare conformazione spaziale, in virtù della quale, con alta probabilità, tutti i punti del Reticolo che cadono nell'intorno di un opportuno punto dello spazio Euclideo  $\mathbb{R}^n$  (dipendente soltanto dai responsi dell'oracolo), esibiscono la stessa struttura, ed in particolare hanno l'ultima componente del tipo  $m + \frac{\gamma}{p}$ , con  $m \in \mathbb{Z}$ . Adoperando un algoritmo di Approssimazione, quale l'algoritmo LLL, è possibile trovare in tempo polinomiale uno di tali punti del Reticolo, e ricavare da esso il numero nascosto  $\gamma$ .

La probabilità con cui l'algoritmo ha successo dipende dalla distribuzione delle potenze del tipo  $\alpha^{x_i} \bmod p$  all'interno di  $\mathbb{Z}_p^*$ . Nell'esposizione presentata in [BV96], si considera soltanto il caso in cui  $\alpha$  sia un generatore di  $\mathbb{Z}_p^*$ , per cui alcune semplici considerazioni combinatorie sono sufficienti per descrivere tale distribuzione. Tuttavia, per talune interessanti applicazioni di tale soluzione (fra le quali la  $k$ -sicurezza del segreto Diffie-Hellman), sarebbe utile poter applicare il risultato ottenuto anche nel caso che  $\alpha$  sia un elemento di ordine  $T < p-1$ . In [VS00], Vasco e Shparlinski mostrano come in effetti sia possibile "rilassare" la condizione sull'ordine di  $\alpha$ , conferendo così maggiore generalità all'algoritmo descritto in [BV96].

Prima di passare alla dimostrazione delle garanzie di successo dell'algoritmo menzionato sopra, è necessario introdurre un lemma, dimostrato in [VS00], che consente di stimare il numero  $N_{\gamma, \alpha, p}(r, h)$  di elementi del tipo  $\gamma \alpha^x \bmod p$  che cadono nell'intervallo  $[r, r+h-1] \subset \mathbb{Z}_p^*$ .

**Lemma 4.1.** *Per qualunque  $\epsilon > 0$ , esiste un  $\delta > 0$  tale che, per ogni elemento  $\alpha$  di ordine  $T \geq p^{1/3+\epsilon}$ , risulta che:*

$$\max_{0 \leq r, h \leq p-1} \max_{\gcd(\gamma, p)=1} |N_{\gamma, \alpha, p}(r, h) - T \cdot h/p| = \mathcal{O}(T^{1-\delta})$$

*Dimostrazione.* Per la dimostrazione si rimanda a [VS00]. □

In altri termini, il Lemma 4.1 afferma che, per gli elementi di  $\mathbb{Z}_p^*$  di ordine non troppo piccolo, il numero medio di potenze di  $\alpha$  che cadono in un dato intervallo, non si discosta molto dal valore  $h \cdot \frac{T}{p}$ , che si ottiene con una semplice analisi del tipo “casi favorevoli su casi possibili”.

Al fine di semplificare l'esposizione che segue, si denota con  $\mathcal{L}_{\alpha,p}(\xi_1, \dots, \xi_d)$  il Reticolo generato dalla base i cui vettori sono le righe della matrice seguente:

$$\begin{pmatrix} p & 0 & \dots & 0 & 0 \\ 0 & p & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & p & 0 \\ \xi_1 & \xi_2 & \dots & \xi_d & 1/p \end{pmatrix}$$

Si considera adesso il seguente lemma, che fornisce una stima della probabilità che l'algoritmo termini con successo:

**Lemma 4.2.** *Siano fissati due numeri  $d$  e  $\mu$  ed un intero  $\gamma \in \mathbb{Z}_p^*$ . Per ogni  $\epsilon > 0$ ,  $p$  sufficientemente grande, ed ogni elemento  $\alpha \in \mathbb{Z}_p^*$  di ordine  $T \geq p^{1/3+\epsilon}$ , risulta che, scegliendo uniformemente  $x_1, \dots, x_d \in \{0, \dots, T-1\}$ , la probabilità che, comunque si scelga  $\beta \not\equiv \gamma \pmod{p}$ , per almeno uno degli  $x_i$  accada che*

$$\text{dist}_p(\beta\alpha^{x_i}, \gamma\alpha^{x_i}) > p2^{-\mu+1}$$

è più che  $1 - 2^{-\sqrt{n}}$ .

*Dimostrazione.* Definiamo

$$\begin{aligned} \text{dist}_p(\beta_1, \beta_2) &\doteq \min_{b \in \mathbb{Z}} |\beta_1 - \beta_2 - bp| \\ &= \min\{|\beta_1 \bmod p - \beta_2 \bmod p|, p - |\beta_1 \bmod p - \beta_2 \bmod p|\} \end{aligned}$$

Sia  $x$  un intero scelto uniformemente in  $\{0, \dots, T-1\}$ . Dal Lemma 4.1, segue che per ogni  $\beta_1$  e  $\beta_2$  per cui  $\beta_1 \not\equiv \beta_2 \pmod{p}$ , detta  $Pr(\beta_1, \beta_2)$  la probabilità dell'evento:

$$\mathcal{E}(\beta_1, \beta_2) = \text{“ } \text{dist}_p(\beta_1 \alpha^x, \beta_2 \alpha^x) > p2^{-\mu+1} \text{”}$$

risulta che, per qualche  $\delta > 0$ :

$$Pr(\beta_1, \beta_2) = 1 - 2^{-\mu+2} + \mathcal{O}(T^{-\delta})$$

che, per  $p$  sufficientemente grande, si può riscrivere

$$Pr(\beta_1, \beta_2) \geq 1 - \frac{5}{2^\mu}.$$

Sia  $\beta \in \mathbb{Z}$ ,  $\beta \not\equiv \gamma \pmod{p}$ ; si può quindi applicare la minorazione di cui sopra a  $\beta$  e  $\gamma$ :

$$Pr(\beta, \gamma) = Pr(\mathcal{E}(\beta, \gamma)) \geq 1 - \frac{5}{2^\mu}$$

Quindi:

$$Pr(\mathcal{E}^C(\beta, \gamma)) = Pr(\text{dist}_p(\beta \alpha^x, \gamma \alpha^x) \leq p2^{-\mu+1}) \leq \frac{5}{2^\mu}$$

Dal momento che  $x_1, \dots, x_d$  sono, per ipotesi, scelti uniformemente ed indipendentemente in  $\{0, \dots, T-1\}$ , è possibile applicare ad essi tale risultato:

$$Pr(dist_p(\beta\alpha^{x_i}, \gamma\alpha^{x_i}) \leq p2^{-\mu+1}) \leq \frac{5}{2^\mu}, \quad 1 \leq i \leq d$$

Allora:

$$Pr(\forall i \in [1, d]. \mathcal{E}_i^C(\beta, \gamma)) = Pr(\forall i \in [1, d]. (dist_p(\beta\alpha^{x_i}, \gamma\alpha^{x_i}) \leq p2^{-\mu+1})) \leq \left(\frac{5}{2^\mu}\right)^d$$

Questa relazione vale qualunque sia  $\beta$ , purché  $\beta \not\equiv \gamma \pmod{p}$ .

Per il Principio di Inclusione-Esclusione, la probabilità che per almeno uno dei possibili  $\beta$  l'evento  $\mathcal{E}_i(\beta, \gamma)$  sia falso è:

$$\begin{aligned} Pr(\exists \beta \not\equiv \gamma \pmod{p} : \forall i \in [1, d]. \mathcal{E}_i^C(\beta, \gamma)) = \\ Pr\left(\bigvee_{\beta \not\equiv \gamma \pmod{p}} (\forall i \in [1, d]. \mathcal{E}_i^C(\beta, \gamma) \mid \beta)\right) \\ \leq \sum_{\beta \not\equiv \gamma \pmod{p}} Pr(\forall i \in [1, d]. \mathcal{E}_i^C(\beta, \gamma) \mid \beta) \\ \leq (p-1) \left(\frac{5}{2^\mu}\right)^d \end{aligned}$$

Passando all'evento contrario:

$$\begin{aligned} Pr(\forall \beta \not\equiv \gamma \pmod{p}. \exists i \in [1, d] : \mathcal{E}_i^C(\beta, \gamma)) = \\ Pr(\forall \beta \not\equiv \gamma \pmod{p}. \exists i \in [1, d] : (dist_p(\beta \alpha^{x_i}, \gamma \alpha^{x_i}) > p2^{-\mu+1})) \\ > 1 - (p-1) \left(\frac{5}{2^\mu}\right)^d \end{aligned}$$

Poiché:

$$\begin{aligned} d(\mu - \log 5) &= \lceil \sqrt{n} \rceil \sqrt{n} + 2 \lceil \sqrt{n} \rceil (3 - \log 5) \\ &\geq n + (6 - \log 25) \sqrt{n} > n + (6 - 5) \sqrt{n} \\ &= n + \sqrt{n} = \lceil \log p \rceil + \sqrt{n} \geq \log p + \sqrt{n} \\ &> \log(p-1) + \sqrt{n} \end{aligned}$$

risulta, passando alle potenze in base 2:

$$\left(\frac{2^\mu}{5}\right)^d > (p-1)2^{\sqrt{n}} \Rightarrow (p-1) \left(\frac{5}{2^\mu}\right)^d < 2^{-\sqrt{n}}$$

da cui:

$$1 - (p-1) \left(\frac{5}{2^\mu}\right)^d > 1 - \frac{1}{2^{\sqrt{n}}}$$

In definitiva:

$$Pr(\forall \beta \not\equiv \gamma \pmod{p} \exists i \in [1, d] : dist_p(\beta \alpha^{x_i}, \gamma \alpha^{x_i}) > p2^{-\mu+1}) > 1 - \frac{1}{2^{\sqrt{n}}}$$

Quindi, in corrispondenza di qualunque intero  $\beta$ , il cui valore modulo  $p$  non sia  $\gamma$ , la probabilità che per almeno un  $x_i$  risulti  $\text{dist}_p(\beta\alpha^{x_i}, \gamma\alpha^{x_i}) > p2^{-\mu+1}$  è più che  $1 - 2^{-\sqrt{n}}$ .  $\square$

Questo Lemma è interessante perché, dal momento che la sua tesi vale solo con una certa probabilità (per quanto alta), tutte le conclusioni che da essa possono essere derivate saranno soggette anch'esse alla stessa restrizione probabilistica.

Questo è il caso del Teorema seguente, che garantisce (con alta probabilità) la correttezza del metodo di Approssimazione in un Reticolo per il Problema Randomizzato del Numero Nascosto.

**Teorema 4.3.** *Sotto le ipotesi del Lemma 4.2, posto  $d \doteq \lceil \sqrt{n} \rceil$  e  $\mu \doteq \frac{\sqrt{n}}{2} + 3$ , con probabilità  $P > 1 - 2^{-\sqrt{n}}$ , comunque si prenda un vettore  $\mathbf{u} = (u_1, \dots, u_d, 0)$  tale che:*

$$\sqrt{\sum_{i=1}^d [(\gamma\alpha^{x_i} \bmod p) - u_i]^2} \leq p2^{-\mu}$$

ogni vettore  $\mathbf{v} = (v_1, \dots, v_d, v_{d+1}) \in \mathcal{L}_{\alpha,p}(\xi_1, \dots, \xi_d)$  per cui risulti:

$$\sqrt{\sum_{i=1}^d (v_i - u_i)^2} \leq p2^{-\mu}$$

è della forma  $\mathbf{v} = (\beta\alpha^{x_1} \bmod p, \dots, \beta\alpha^{x_d} \bmod p, \beta/p)$ , per qualche valore  $\beta \equiv \gamma \pmod{p}$ .

*Dimostrazione.* Sia  $\mathbf{u} \equiv (u_1, \dots, u_d, 0)$  un vettore di  $\mathbb{R}^n$  per cui risulti:

$$\sqrt{\sum_{i=1}^d [(\gamma \alpha^{x_i} \bmod p) - u_i]^2} \leq p2^{-\mu} \quad (\dagger)$$

Sia  $\mathbf{v} = (v_1, \dots, v_d, v_{d+1})$  un vettore di  $\mathcal{L}_{\alpha,p}(\xi_1, \dots, \xi_d)$  per cui risulti:

$$\sqrt{\sum_{i=1}^d (v_i - u_i)^2} \leq p2^{-\mu} \quad (\ddagger)$$

Essendo  $\mathbf{v}$  un vettore del Reticolo, esso sarà del tipo

$$\mathbf{v} = (\beta \xi_1 - b_1 p, \dots, \beta \xi_d - b_d p, \beta/p),$$

per qualche  $b_1, \dots, b_d, \beta \in \mathbb{Z}$ .

Si distinguono due casi:

1.  $\beta \equiv \gamma \pmod{p}$

In tal caso, le prime  $d$  componenti di  $\mathbf{v}$  sono pari a  $(\beta \xi_i \bmod p)$ . Infatti:

$$v_i = \beta \xi_i - b_i p \Rightarrow v_i \equiv \beta \xi_i \pmod{p}$$

ed essendo  $\beta \equiv \gamma \pmod{p}$ , si ha

$$v_i \equiv \gamma \xi_i \pmod{p} \Rightarrow v_i = (\gamma \xi_i \bmod p) - b_i^*,$$

per qualche  $b_i^* \in \mathbb{Z}$ . Unendo la  $(\dagger)$  e la  $(\ddagger)$ , si ricava che:

$$\sqrt{\sum_{i=1}^d ((\gamma \xi_i \bmod p) - v_i)^2} \leq p2^{-\mu} + p2^{-\mu} = p2^{-\mu+1}$$

da cui  $|(\gamma\xi_i \bmod p) - v_i| \leq p2^{-\mu+1}$ . Ma

$$|(\gamma\xi_i \bmod p) - v_i| = |(\gamma\xi_i \bmod p) - ((\gamma\xi_i \bmod p) - b_i^*p)| = |b_i^*p| \leq p2^{-\mu+1},$$

ed essendo  $2^{-\mu+1} < 1$ , segue che  $|b_i^*p| < p$ , vale a dire  $b_i^*p$  è un multiplo di  $p$  con valore assoluto più piccolo di quello di  $p$ : l'unico valore possibile è allora  $b_i^* = 0$ , e quindi  $v_i = (\gamma\xi_i \bmod p)$ ,  $1 \leq i \leq d$ .

## 2. $\beta \not\equiv \gamma \pmod{p}$

Sotto questa ipotesi è possibile applicare la tesi del Lemma 4.2, per cui con probabilità  $P$  più che  $1 - 2^{-\sqrt{n}}$ , esiste un  $i^* \in [1, d]$  per cui  $\text{dist}_p(\beta\alpha^{x_{i^*}}, \gamma\alpha^{x_{i^*}}) > p2^{-\mu+1}$ . Si prova ora che ciò costituisce un assurdo.

Si consideri la (§)

$$\begin{aligned} p2^{-\mu} &\geq \sqrt{\sum_{i=1}^d (v_i - u_i)^2} \\ &\geq |v_{i^*} - u_{i^*}| = |\beta\xi_{i^*} - b_{i^*}p - u_{i^*}| \\ &= |\beta\xi_{i^*} - \gamma\xi_{i^*} + \gamma\xi_{i^*} - b_{i^*}p - u_{i^*}| \\ &= |\beta\xi_{i^*} - \gamma\xi_{i^*} + (\gamma\xi_{i^*} \bmod p) + (\gamma\xi_{i^*} \text{div } p) \cdot p - b_{i^*}p - u_{i^*}| \\ &= |\beta\xi_{i^*} - \gamma\xi_{i^*} + ((\gamma\xi_{i^*} \text{div } p) - b_{i^*}) \cdot p - (u_{i^*} - (\gamma\xi_{i^*} \bmod p))| \\ &\geq |\beta\xi_{i^*} - \gamma\xi_{i^*} + ((\gamma\xi_{i^*} \text{div } p) - b_{i^*}) \cdot p| + |u_{i^*} - (\gamma\xi_{i^*} \bmod p)| \\ &\geq \min_{b \in \mathbb{Z}} |\beta\xi_{i^*} - \gamma\xi_{i^*} + bp| + |(\gamma\xi_{i^*} \bmod p) - u_{i^*}| \\ &= \text{dist}_p(\beta\xi_{i^*}, \gamma\xi_{i^*}) + |(\gamma\xi_{i^*} \bmod p) - u_{i^*}| \end{aligned} \quad (\#)$$

Dalla (†) segue ovviamente che  $|(\gamma\xi_{i^*} \bmod p) - u_{i^*}| \leq p2^{-\mu}$ , da cui:

$$= \text{dist}_p(\beta\alpha^{x_{i^*}}, \gamma\alpha^{x_{i^*}}) - p2^{-\mu} > p2^{-\mu+1} - p2^{-\mu} = p2^{-\mu}$$

cioè  $p2^{-\mu} > p2^{-\mu}$ . Si è giunti ad un assurdo, che prova che il caso 2 non può aversi (se non con probabilità minore di  $2^{-\sqrt{n}}$ ).

Ne segue pertanto che l'ultima componente del vettore  $\mathbf{v}$  è del tipo  $\beta/p$ , con  $\beta \equiv \gamma \pmod{p}$ , e che tutte le altre componenti sono del tipo  $\beta\xi_i \bmod p$   $\square$

#### 4.1.4 Applicazione alla Bit Security della funzione DH

Si considera adesso come sia possibile ricondurre l'analisi di sicurezza dei  $k = \lceil \sqrt{n} \rceil + \lceil \log n \rceil$  bits più significativi del segreto Diffie-Hellman al Problema Randomizzato del Numero Nascosto. Si osservi preliminarmente che la questione in esame può essere riformulata nei seguenti termini: supposto che la funzione  $\text{DH}(\alpha^a, \alpha^b)$  sia difficile da calcolare, allora la funzione  $\text{MSB}_k(\text{DH}(\alpha^a, \alpha^b))$  è anch'essa difficile da calcolare.

Si supponga di disporre di un algoritmo  $\mathcal{A}$  che computi i primi  $k$  bits del segreto (nel senso della Definizione 4.1):

$$\mathcal{A}(\alpha^a \bmod p, \alpha^b \bmod p) = \text{MSB}_k(\text{DH}(\alpha^a \bmod p, \alpha^b \bmod p))$$

A partire da esso vogliamo costruire un oracolo per la versione randomizzata del problema del Numero Nascosto, in cui la quantità  $\gamma$  da scoprire sia  $\alpha^{ab} \bmod p$ , e l'elemento<sup>1</sup>  $\beta$  utilizzato per creare le maschere da moltiplicare per  $\gamma$  sia pari a  $\alpha^b \bmod p$ .

Come già detto in precedenza, in tale versione del problema, l'oracolo, preso in input  $x$ , restituisce i  $k$  bits più significativi del prodotto di  $\gamma$  per una potenza di  $\beta$ :

$$\mathcal{O}_{\gamma, \beta}(x) = \text{MSB}_k(\gamma \cdot \beta^x \bmod p)$$

Ma allora risulta:

$$\begin{aligned} \mathcal{O}_{\gamma, \beta}(x) &= \text{MSB}_k(\gamma \cdot \beta^x \bmod p) = \text{MSB}_k(\alpha^{ab} \bmod p \cdot (\alpha^b)^x \bmod p) \\ &= \text{MSB}_k(\alpha^{(a+x)b} \bmod p) = \text{MSB}_k(\text{DH}(\alpha^{a+x} \bmod p, \alpha^b \bmod p)) \\ &= \mathcal{A}(\alpha^{a+x} \bmod p, \alpha^b \bmod p) \end{aligned}$$

Di conseguenza, è possibile utilizzare l'algoritmo  $\mathcal{A}$  per ottenere un qualunque numero di coppie  $(\beta^{x_i} \bmod p, \mathcal{O}_{\gamma, \beta}(x_i)) \equiv (\xi_i, \mathcal{O}_{\gamma}(\xi_i))$  dove l'equivalenza deriva dalle relazioni  $\xi_i = \beta^{x_i} \bmod p$  e  $\mathcal{O}_{\gamma}(\xi_i) = \mathcal{O}_{\gamma, \beta}(x_i)$ .

Effettuando  $2\lceil\sqrt{n}\rceil$  di tali queries, si ottengono informazioni sufficienti per

---

<sup>1</sup>Nelle sezioni precedenti, tale quantità è stata denotata con  $\alpha$ , ed era richiesto che si trattasse di un generatore; in virtù del Lemma 4.3 sopra trattato, è possibile estendere la scelta ad un qualunque elemento  $\beta$  di ordine  $T$  non troppo piccolo, e si è preferito quindi cambiare notazione.

scoprire il Numero Nascosto dell'oracolo, utilizzando la tecnica illustrata nella sezione precedente. Ma, per come è stato costruito l'oracolo, il Numero Nascosto  $\gamma$  è proprio  $\alpha^{ab} \bmod p = \text{DH}(\alpha^a \bmod p, \alpha^b \bmod p)$ . Di conseguenza, esiste una riduzione polinomiale dal problema di determinare i  $k$  bits più significativi del segreto Diffie-Hellman al problema di determinare per intero tale segreto.

Ciò prova, a norma della Definizione 2.3, che i primi  $\lceil \sqrt{n} \rceil + \lceil \log n \rceil$  sono sicuri per la funzione Diffie-Hellman.

L'utilità pratica di un tale risultato è legata alla discussione fatta nella Sezione 3.3.1, a proposito della necessità di ricavare una chiave di sessione corta dalla chiave a 1024 bits ottenuta per mezzo del Protocollo Diffie-Hellman. Applicando il risultato appena mostrato per un primo  $p$  a 1024 bits, è possibile garantire che utilizzare più di 42 bits dell'intera chiave è sicuro sotto l'assunzione CDH.

Un'ultima osservazione relativa al risultato appena presentato riguarda la tecnica utilizzata per derivarlo. Si tratta di uno dei pochi casi in cui gli strumenti della Teoria dei Numeri, e in particolare il Concetto di Reticolo, vengono utilizzati al fine di provare la robustezza di un sistema crittografico, piuttosto che per dimostrarne l'inconsistenza, realizzando ingegnosi attacchi crittoanalitici.

## 4.2 Sicurezza di bits interni della funzione DH

In questa sezione si passa a considerare la possibilità di generalizzare il risultato di [BV96] al caso di  $k$  bits consecutivi ma interni al segreto Diffie-Hellman. Tale estensione costituisce il contributo originale del presente lavoro di Tesi.

La difficoltà di estendere tale risultato sta nell'impossibilità di generalizzare la notazione di bits più significativi introdotta nella Definizione 4.1. Mentre i  $k$  bits più significativi individuano un intervallo di valori contigui, nel caso generale resta individuata l'unione di tanti spezzoni di  $\mathbb{Z}_p^*$ . È pertanto più opportuno utilizzare la nozione tradizionale di bits, il che, tra l'altro, non presenta i problemi discussi introducendo la notazione di Micali-Goldwasser, in quanto non si ha a che fare con gli ultimi bits.

Nonostante questa variazione, la riduzione polinomiale dal problema di determinare una finestra di  $k$  bits interni della funzione Diffie-Hellman al problema di calcolarne per intero il segreto, viene realizzata facendo uso di una generalizzazione del problema del Numero Nascosto.

### 4.2.1 Hidden Number Problem per $k$ bits interni

Sia  $p$  un numero primo e sia  $n \doteq \lceil \log p \rceil$ . Sia  $\gamma \in \mathbb{Z}_p^*$  un elemento “nascosto”, da indovinare utilizzando un oracolo che fornisce i bits alle posizioni  $s, s+1, \dots, s+w+1$  di  $\gamma \cdot \alpha^x \pmod p$ , per  $x \in \mathbb{Z}_{p-1}$ . Si denoti con  $\xi$  la potenza  $x^{esima}$  di  $\alpha$ , cioè  $\xi = \alpha^x \pmod p$ . Per descrivere l’output dell’oracolo, si noti che  $\gamma \cdot \xi \pmod p$  si può esprimere come:

$$\gamma \cdot \xi \pmod p = x^{(1)} + x^{(2)} \cdot 2^s + x^{(3)} \cdot 2^{s+w}$$

per qualche  $x^{(1)} < 2^s$ ,  $x^{(2)} < 2^w$  e  $x^{(3)} < p \cdot 2^{-w-s}$ . Il responso dell’oracolo si può quindi scrivere come:

$$\mathcal{O}_\gamma(\xi) = x^{(2)} \cdot 2^s$$

È dunque possibile considerare la differenza:

$$\gamma \cdot \xi \pmod p - \mathcal{O}_\gamma(\xi) = x^{(1)} + 2^{w+s} \cdot x^{(3)} \quad (*)$$

I possibili valori che questa differenza può assumere sono  $2^s \cdot p2^{-s-w} = p2^{-w}$ , ma risultano disseminati in un intervallo molto più ampio, che praticamente coincide con  $\mathbb{Z}_p^*$ . Si nota però, che la distribuzione di tali valori presenta dei grossi buchi, di dimensione  $2^w$ . Di conseguenza, si potrebbe tentare di “accorpare” i vari segmenti, ottenendo così per la differenza sopra evidenziata,

un campo di variabilità più ridotto. A tale scopo, nel seguito, si utilizzano alcuni artifici introdotti in [FHK<sup>+</sup>88].

Prima di affrontare tale problema, si introduce in  $\mathbb{Z}_p^*$ , una generalizzazione del “valore assoluto”. Si immagini di disporre gli elementi di  $\mathbb{Z}_p^*$  lungo una circonferenza e, dato un valore  $x$ , si consideri il minimo numero di elementi da attraversare per raggiungere l’origine, ruotando in senso orario o antiorario. Questo numero è il *valore assoluto modulare* di  $x$ , denotato con  $|x|_p$ . Dalla definizione si nota che risulta:

$$|x|_p \doteq \min\{|x| \bmod p, p - (|x| \bmod p)\}, \forall x \in \mathbb{Z}$$

Tale nozione è strettamente legata alla distanza  $dist_p$  già introdotta:

$$\begin{aligned} dist_p(x, y) &\doteq \min_{b \in \mathbb{Z}} |x - y - bp| \\ &= \min\{|x \bmod p - y \bmod p|, p - |x \bmod p - y \bmod p|\} \\ &= |x - y|_p \end{aligned}$$

Il valore assoluto modulare gode delle seguenti proprietà:

$$\begin{aligned} |x + y|_p &\leq |x|_p + |y|_p \\ (|x - y|_p)^2 &= (x \bmod p - y \bmod p)^2 \\ (|x|_p \cdot |y|_p) \in \mathbb{Z}_p &\Rightarrow |xy|_p = |x|_p \cdot |y|_p \end{aligned}$$

Nel contesto che si sta esaminando, le nozioni di “valore assoluto modulare” e distanza risultano molto utili; in particolare, il valore assoluto modulare è assimilabile ad una norma (benché a rigore non lo sia), in quanto fornisce una stima della “grandezza” di un intero relativo, quando viene ridotto modulo  $p$ .

Ritornando al problema del Numero Nascosto, si vuole modificare la relazione (\*), moltiplicandola per una quantità opportuna, in modo che al primo membro si ottenga una quantità i cui possibili valori siano compresi in un unico intervallo di dimensione piccola, ovvero una quantità variabile la cui norma risulti piccola.

Si denoti con  $a$  tale valore incognito; si ottiene:

$$a(\gamma \cdot \xi \bmod p) - a\mathcal{O}_\gamma(\xi) = ax^{(1)} + 2^{w+s}a \cdot x^{(3)}$$

$$\Downarrow$$

$$\text{dist}_p(a(\gamma \cdot \xi \bmod p), a\mathcal{O}_\gamma(\xi)) = |ax^{(1)} + 2^{w+s}a \cdot x^{(3)}|_p$$

$$\Downarrow$$

$$\text{dist}_p(\gamma \cdot a\xi \bmod p, a\mathcal{O}_\gamma(\xi)) = |ax^{(1)} + 2^{w+s}a \cdot x^{(3)}|_p$$

Si ponga adesso  $z \doteq ax^{(1)} + 2^{w+s}a \cdot x^{(3)}$ , e  $\tilde{a} \doteq a\mathcal{O}_\gamma(\xi)$ . Allora:

$$\text{dist}_p(\gamma \cdot a\xi \bmod p, \tilde{a}) = |z|_p$$

La norma di  $z$ , per le proprietà viste prima, può essere riscritta come:

$$|z|_p \leq |ax^{(1)}|_p + |a2^{w+s}x^{(3)}|_p = |a|_p|x^{(1)}|_p + |a2^{w+s}|_p|x^{(3)}|_p \quad (**)$$

Dal momento che  $x^{(1)} < 2^s$  e  $x^{(3)} < p \cdot 2^{-w-s}$ , per minimizzare la norma di  $z$  si necessita di un valore  $a \in \mathbb{Z}_p^*$  con le seguenti due caratteristiche:

$$\begin{cases} |a|_p \leq p \cdot 2^{-s-w/2} \\ |a \cdot 2^{s+w}|_p \leq 2^{s+w/2} \end{cases}$$

Determinare un elemento di  $\mathbb{Z}_p^*$  che soddisfi i vincoli esposti è possibile, in tempo polinomiale,<sup>2</sup> riconducendo i calcoli ad un problema di Programmazione

Intera a tre dimensioni nelle variabili  $a, y_1, y_2$ :

$$\begin{cases} -p \cdot 2^{-s-w/2} \leq a - y_1p \leq p \cdot 2^{-s-w/2} \\ -2^{s+w/2} \leq a \cdot 2^{s+w} - y_2p \leq 2^{s+w/2} \\ 0 < a < p \end{cases}$$

Per tale scelta di  $a$ , dalla (\*\*) si ottiene che:

$$|z|_p < p \cdot 2^{-s-w/2} \cdot 2^s + 2^{s+w/2} \cdot p \cdot 2^{-s-w} = p \cdot 2^{-w/2} + p \cdot 2^{-w/2} = p \cdot 2^{-w/2+1}$$

In definitiva:

$$\text{dist}_p(\gamma \cdot a\xi \pmod{p, \tilde{a}}) < p \cdot 2^{-w/2+1}$$

---

<sup>2</sup>Nell'Appendice A è brevemente descritto un algoritmo efficiente, dovuto a H. Lenstra [Len83], per risolvere simili problemi di Programmazione Intera.

La relazione ottenuta al termine di tale manipolazione mostra che il valore  $\tilde{a} \doteq a\mathcal{O}_\gamma(\xi) \bmod p$ , noto poiché prodotto del responso dell'oracolo per la soluzione del Programma Intero presentato in precedenza, risulta essere sufficientemente prossimo ad un multiplo del numero incognito  $\gamma$ .

Per sfruttare l'esistenza di tali valori per scoprire il Numero Nascosto si fa uso delle stesse tecniche di approssimazione nei Reticoli, illustrate in precedenza per il caso dei  $k$  bits iniziali.

L'input del problema è costituito da una lista di  $d$  coppie del tipo  $(\xi_i, \mathcal{O}_\gamma(\xi_i))$ . Si ponga  $\tilde{a}_i \doteq a\mathcal{O}_\gamma(\xi_i)$ ,  $1 \leq i \leq d$ ; per quanto osservato in precedenza, risulta che  $dist_p(\gamma \cdot a\xi_i \bmod p, \tilde{a}_i) < p2^{-w/2+1}$ .

Si consideri a questo punto il Reticolo  $\mathcal{L}_{\alpha,p}(a\xi_1, \dots, a\xi_d)$  a  $d+1$  dimensioni generato dalle righe della matrice:

$$\begin{pmatrix} p & 0 & \dots & 0 & 0 \\ 0 & p & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & p & 0 \\ a\xi_1 & a\xi_2 & \dots & a\xi_d & 1/p \end{pmatrix}$$

Il generico vettore di tale Reticolo può essere espresso come:

$$(\beta a \xi_1 - b_1 p, \dots, \beta a \xi_d - b_d p, \beta/p) \quad \text{con } \beta, b_1, \dots, b_d \in \mathbb{Z}$$

Quando i coefficienti sono del tipo

$$\bar{\beta} \equiv \gamma \pmod{p} \quad \bar{b}_i \doteq \beta a \xi_i \operatorname{div} p, \quad 1 \leq i \leq d,$$

si ottengono dei punti del Reticolo speciali  $\mathbf{v}_\beta$ , in cui l'ultima componente è pari a  $\beta/p$ , mentre tutte le altre sono del tipo  $(\gamma a \xi_i \operatorname{mod} p)$ .

Si consideri adesso, nello spazio Euclideo a  $d + 1$  dimensioni, il punto  $\mathbf{u} \equiv (\tilde{a}_1, \dots, \tilde{a}_d, 0)$ , in cui le prime  $d$  componenti sono le informazioni  $\tilde{a}_i$  ricavabili a partire dai responsi ottenuti dall'oracolo  $\mathcal{O}_\gamma$ . I punti  $\mathbf{u}$  e  $\mathbf{v}_\alpha$  risultano essere fra di loro vicini; la loro distanza è:

$$\begin{aligned} \|\mathbf{v}_\alpha - \mathbf{u}\| &= \sqrt{((\gamma a \xi_1 \operatorname{mod} p) - \tilde{a}_1)^2 + \dots + ((\gamma a \xi_d \operatorname{mod} p) - \tilde{a}_d)^2 + (\gamma/p)^2} \\ &= \sqrt{(\operatorname{dist}_p(\gamma a \xi_1 \operatorname{mod} p, \tilde{a}_1))^2 + \dots + (\operatorname{dist}_p(\gamma a \xi_d \operatorname{mod} p, \tilde{a}_d))^2 + (\alpha/p)^2} \\ &< \sqrt{(d+1)(p2^{-w/2+1})^2} < \sqrt{d+1} \cdot p2^{-w/2+1} \end{aligned}$$

Ciò implica che il punto del Reticolo che meglio approssima  $\mathbf{u}$  dista da esso meno di  $\sqrt{d+1} \cdot p2^{-w/2+1}$ .

Eseguendo l'algoritmo di Babai [Bab86] per la determinazione di un punto di  $\mathcal{L}$  che approssimi  $\mathbf{u}$ , si otterrà, in tempo polinomiale, un punto  $\mathbf{w}$  del

Reticolo, molto prossimo ad  $\mathbf{u}$ :

$$\|\mathbf{w} - \mathbf{u}\| \leq 2^{\frac{d}{4}} \min\{\|\mathbf{z} - \mathbf{u}\| : \mathbf{z} \in \mathcal{L}\} \leq 2^{\frac{d}{4}} \cdot \sqrt{d+1} \cdot p2^{-w/2+1}$$

In forza del Teorema 4.3, se tale punto è sufficientemente vicino a  $\mathbf{u}$ , esso sarà del tipo  $\mathbf{v}_\beta$ . In definitiva, seguendo tale procedimento, si è ricavato un punto la cui ultima componente è  $\beta/p$ , con  $\beta \equiv \gamma \pmod{p}$ ; a questo punto è sufficiente prendere la parte decimale di tale numero, e moltiplicarla per  $p$ , per recuperare il numero “nascosto”  $\gamma$ .

Per soddisfare i requisiti del Teorema 4.3, è necessario che  $w$ , cioè la quantità di informazione contenuta in ogni singolo responso dell’oracolo  $\mathcal{O}_\gamma$ , sia circa uguale a  $2\sqrt{n} + \log n$ . In altre parole, la quantità di informazioni necessarie ad ogni singola queries per risolvere il problema randomizzato del numero nascosto, è, nel caso che la finestra di bits sia interna, circa il doppio rispetto al caso in cui si tratta dei bits più significativi.

A questo punto, è possibile dimostrare che la finestra di bits dalla posizione  $s$  alla posizione  $s + w - 1$  risultano sicuri nel senso della Definizione 2.3. In particolare, è possibile effettuare esattamente la stessa riduzione polinomiale analizzata nella Sezione 4.1.4: ne segue la sicurezza dei bits alle posizioni  $s, s + 1, \dots, s + w - 1$ .

### 4.3 Conclusione

L'utilizzo di opportune tecniche di approssimazione all'interno dei Reticoli si è rivelato uno strumento assai flessibile per l'analisi della Bit Security della funzione Diffie-Hellman.

Al risultato presentato in [BV96], viene ad affiancarsi l'estensione, elaborata in questa tesi, al caso di una finestra di bits contigui ovunque collocati.

La tecnica proposta per la compattazione dell'insieme di variabilità di un valore, del quale siano noti soltanto i bits in posizioni assegnate, può verosimilmente essere generalizzata per gestire anche lo scenario con più di una finestra di bits, ovvero nel caso in cui i  $k$  bits in esame non siano tutti contigui.

Sarebbe interessante vedere come cresce la quantità di informazione necessaria per determinare univocamente il Numero Nascosto, al crescere del numero di finestre.

Un ulteriore problema aperto, che sembra comunque ben lontano da una rapida soluzione, è quello di stabilire se le tecniche legate ai Reticoli possano consentire di provare la sicurezza di un singolo bits (il MSB o il LSB, ad esempio) del segreto Diffie-Hellman, risolvendo così la questione dell'esistenza di un *hardcore bit* per la funzione Diffie-Hellman.

# Appendice A

## Approssimazione nei Reticoli

*Gran parte delle tecniche per costruire ed attaccare le primitive crittografiche, esaminate in questa Tesi, sono ispirate a problemi di Matematica Discreta. Per tale ragione, in appendice si presentano brevemente alcune nozioni di Geometria dei Numeri, ed in particolare il concetto di Reticolo, che consente di risolvere, in maniera efficiente, talune questioni di sicurezza che sorgono nell'ambito della Crittografia a Chiave Pubblica.*

## A.1 Spazi Vettoriali

I Reticoli sono delle strutture matematiche molto utilizzate in *Teoria dei Numeri*. La loro definizione è strettamente legata al concetto di *Spazio Vettoriale*, che viene ripreso di seguito per completezza.

**Definizione A.1 (Spazio Vettoriale).** Dato un campo  $\mathbb{F}$ , si chiama *Spazio Vettoriale* la terna  $(\mathcal{V}, \oplus, \odot)$ , dove  $\mathcal{V}$  è un insieme i cui elementi sono detti *vettori*,  $\oplus : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V}$  è un'operazione detta *somma vettoriale*, e  $\odot : \mathbb{F} \times \mathcal{V} \rightarrow \mathcal{V}$  è un'operazione detta *prodotto scalare*, e inoltre risulta che:

1.  $\mathcal{V}$  è un gruppo abeliano rispetto alla somma vettoriale  $\oplus$ ;
2. Il prodotto scalare  $\odot$  soddisfa la relazione:

$$\forall a, b \in \mathbb{F} \quad \forall \mathbf{v} \in \mathcal{V} \quad a \odot (b \odot \mathbf{v}) = (ab) \odot \mathbf{v}$$

3. La somma vettoriale  $\oplus$  si distribuisce rispetto al prodotto scalare  $\odot$ :

$$\forall a \in \mathbb{F} \quad \forall \mathbf{v}, \mathbf{w} \in \mathcal{V} \quad a \odot (\mathbf{v} \oplus \mathbf{w}) = a \odot \mathbf{v} \oplus a \odot \mathbf{w}$$

Una nozione importante riguardo ai vettori è quella di *base*. Per definirla, è necessario introdurre i concetti di *dipendenza* e *indipendenza lineare*.

**Definizione A.2 (Dipendenza Lineare).** I vettori  $\mathbf{v}_1, \dots, \mathbf{v}_k$  si dicono *linearmente dipendenti* se esiste una loro combinazione lineare, a coefficienti non tutti nulli, che risulta essere nulla:

$$\exists a_1, \dots, a_k \in \mathbb{R}, \max_{1 \leq i \leq k} |a_i| > 0 : \sum_{i=1}^k a_i \cdot \mathbf{v}_i = \mathbf{0}$$

Si noti che nell'ultima espressione si è denotato con  $\mathbf{0}$  il vettore le cui componenti sono tutte nulle. Un modo equivalente di formulare la dipendenza lineare è richiedere che uno dei  $k$  vettori sia esprimibile come combinazione lineare dei restanti  $k - 1$ .

**Definizione A.3 (Indipendenza Lineare).** I vettori  $\mathbf{v}_1, \dots, \mathbf{v}_k$  si dicono *linearmente indipendenti* se *non* esiste alcuna loro combinazione lineare, a coefficienti non tutti nulli, che risulta essere nulla:

$$\forall a_1, \dots, a_k \in \mathbb{R}, \max_{1 \leq i \leq k} |a_i| > 0 \quad \sum_{i=1}^k a_i \cdot \mathbf{v}_i \neq \mathbf{0}$$

**Definizione A.4.** Dati i vettori  $(\mathbf{v}_1, \dots, \mathbf{v}_k)$ , si chiama *sottospazio generato* da essi lo spazio vettoriale  $(\mathcal{V}', \oplus, \odot)$ , dove  $\mathcal{V}'$  è l'insieme dei vettori di  $\mathcal{V}$  che possono essere espressi come combinazione lineare dei  $\mathbf{v}_i$ :

$$\mathcal{V}' \doteq \left\{ \mathbf{v}' = \sum_{i=1}^k a_i \cdot \mathbf{v}_i, \quad a_1, \dots, a_k \in \mathbb{F} \right\}$$

Un insieme di vettori per cui il sottospazio generato è l'intero spazio vettoriale di partenza è detto *insieme di generatori*.

**Definizione A.5 (Base di uno Spazio Vettoriale).** Un insieme di vettori linearmente indipendenti che generano lo spazio vettoriale è detto *base* dello spazio stesso.

Ogni spazio vettoriale ammette almeno una base, e se questa è finita, allora ogni base ha lo stesso numero di elementi, che prende il nome di *dimensione* dello spazio vettoriale.

Un esempio molto comune di spazio vettoriale è lo Spazio Euclideo ad  $n$  dimensioni, in cui  $\mathcal{V} = \mathbb{R}^n$ , e l'operazione di somma vettoriale  $+$  e prodotto scalare  $\cdot$  sono definite come segue:

$$\mathbf{x} + \mathbf{y} = (x_1, \dots, x_n) + (y_1, \dots, y_n) \doteq (x_1 + y_1, \dots, x_n + y_n)$$

$$a \cdot \mathbf{x} = a \cdot (x_1, \dots, x_n) \doteq (ax_1, \dots, ax_n)$$

Nel caso di  $\mathbb{R}^n$ , la dimensione è  $n$ : una sua base (detta *base canonica*) è:

$$((1, 0, \dots, 0), (0, 1, \dots, 0), \dots, (0, 0, \dots, 1))$$

Un'altra operazione che viene spesso introdotta negli spazi vettoriale è il *prodotto interno*  $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{F}$ . Nel caso dello spazio Euclideo ad  $n$  dimensioni, il prodotto interno è definito come:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle (x_1, \dots, x_n), (y_1, \dots, y_n) \rangle \doteq x_1y_1 + \dots + x_ny_n$$

Dato uno spazio vettoriale  $(\mathcal{V}, \oplus, \odot)$ , sorge spesso la domanda di valutare in qualche modo la “grandezza” dei suoi elementi, o di stimare quanto due vettori siano “distanti”. Per dare risposta a tali domande, risultano utili i concetti di *norma* e *distanza*:

**Definizione A.6 (Norma).** Una norma per uno spazio vettoriale  $(\mathcal{V}, \oplus, \odot)$  è una funzione  $\|\cdot\| : \mathcal{V} \rightarrow \mathbb{R}$  tale che:

1.  $\|\mathbf{x}\| \geq 0, \quad \forall \mathbf{x} \in \mathcal{V};$
2.  $\|\mathbf{x}\| = 0 \iff \mathbf{x} = \mathbf{0};$
3.  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{V}$

**Definizione A.7 (Distanza).** Dato uno spazio vettoriale  $(\mathcal{V}, \oplus, \odot)$ , si definisce distanza una funzione  $dist : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$  tale che:

1.  $dist(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y};$
2.  $dist(\mathbf{x}, \mathbf{z}) \leq dist(\mathbf{x}, \mathbf{y}) + dist(\mathbf{z}, \mathbf{y}), \quad \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{V}$

Dai due assiomi della distanza segue anche che essa risulta sempre non negativa e simmetrica:

$$dist(\mathbf{x}, \mathbf{y}) \geq 0, \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{V}$$

$$dist(\mathbf{x}, \mathbf{y}) = dist(\mathbf{y}, \mathbf{x}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{V}$$

Si osservi che ogni norma induce una distanza:  $dist(\mathbf{x}, \mathbf{y}) \doteq \|\mathbf{x} - \mathbf{y}\|$ ; per tale motivo, lavorando in uno spazio vettoriale, spesso si definisce solo la norma, assumendo implicitamente come distanza quella da essa indotta.

In  $\mathbb{R}^n$ , una norma molto usata è la cosiddetta *norma Euclidea*:

$$\|\mathbf{x}\| \doteq \sqrt{x_1^2 + \dots + x_n^2}$$

Si tratta, in sostanza, della radice quadrata del prodotto interno del vettore  $\mathbf{x}$  con se stesso. La distanza indotta da tale norma è detta *distanza Euclidea*:

$$dist(\mathbf{x}, \mathbf{y}) \doteq \sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$$

In questo caso, si può pensare alla norma di un punto come alla sua distanza dall'origine, ovvero si può interpretare la norma di un vettore come la sua "lunghezza". D'altra parte, la distanza fra due punti, definita nello spazio Euclideo come "grandezza" della loro differenza, indica quanto essi siano "vicini", in un senso del tutto analogo a quello dello spazio a tre dimensioni.

## A.2 Reticoli e Forme Ridotte della Base

Nella definizione di Spazio Vettoriale (Definizione A.1), il campo  $\mathbb{F}$  viene talvolta detto *supporto* dello spazio. Se in tale definizione si sostituisce il campo  $\mathbb{F}$  con un anello arbitrario  $\mathbb{A}$ , si ottiene il concetto di *modulo* di supporto  $\mathbb{A}$ .

**Definizione A.8 (Reticolo).** Si definisce Reticolo di Punti  $\mathcal{L}$  (o semplicemente Reticolo) un modulo avente come supporto l'anello  $\mathbb{Z}$  degli interi relativi. Gli elementi del Reticolo  $\mathcal{L}$  vengono detti *punti* del Reticolo.

Nel caso dello spazio Euclideo  $\mathbb{R}^n$ , i Reticoli vengono solitamente introdotti in maniera leggermente diversa, in termini di una base di  $\mathbb{R}^n$ .

**Definizione A.9 (Reticolo generato da una Base).**

Data una base  $B = (\mathbf{b}_1, \dots, \mathbf{b}_n)$  di  $\mathbb{R}^n$ , il Reticolo  $\mathcal{L}(B)$  descritto (o generato) da  $B$  è l'insieme di tutte le possibili combinazioni lineari a coefficienti interi degli elementi di  $B$ :

$$\mathcal{L}(B) \doteq \left\{ \mathbf{x} = \sum_{i=1}^k z_i \mathbf{b}_i, \quad \text{con } z_i \in \mathbb{Z}, 1 \leq i \leq n \right\}$$

Si noti che, mentre considerando le combinazioni a coefficienti reali dei vettori di una base  $B$  è possibile esprimere qualunque vettore  $n$ -dimensionale, se si considerano solo le combinazioni lineari a coefficienti interi, si ottiene un

insieme (infinito) di punti sparpagliati nello spazio Euclideo, disposti secondo una struttura geometrica che dipende dalla base in questione.<sup>1</sup> Per tale ragione, la branca dell'algebra che si occupa dei Reticoli viene anche detta *Geometria dei Numeri* (in contrapposizione alla *Geometria Lineare*).

Ci si potrebbe a questo punto chiedere se ogni base  $B$  genera un Reticolo  $\mathcal{L}$  diverso, o se invece esistono Reticoli che ammettono più di una descrizione. In effetti, per ogni Reticolo esistono infinite basi che lo descrivono.

A questo punto, il quesito interessante diventa: dato un Reticolo  $\mathcal{L}$ , esiste una base che lo descrive meglio delle altre? La risposta a questa domanda potrebbe essere espressa come “sì, e più di una”. Questo perché si possono introdurre diversi criteri di ottimalità, a seconda degli aspetti che si vogliono descrivere. Per tale ragione, si preferisce usare il termine “base ridotta” piuttosto che “base ottimale”.

Una delle più comuni definizioni di “base ridotta” è dovuta a Minkowski: essa fa uso della nozione di norma, introdotta nella Definizione A.6, ed applicabile anche ad un Reticolo  $\mathcal{L}$ , in quanto i suoi punti sono anche punti dello spazio Euclideo  $\mathbb{R}^n$ .

---

<sup>1</sup>Il fatto che una base  $B$  caratterizzi in maniera così netta l'insieme dei punti che appartengono al Reticolo  $\mathcal{L}$  indotto, viene solitamente espresso dicendo che  $B$  *descrive*  $\mathcal{L}$ .

**Definizione A.10 (Base Ridotta secondo Minkowski).**

Una base  $B = (\mathbf{b}_1, \dots, \mathbf{b}_n)$  è ridotta secondo Minkowski se:

1.  $\mathbf{b}_1$  è il più corto vettore non nullo presente in  $\mathcal{L}$ ;
2. Per  $2 \leq i \leq n$ ,  $\mathbf{b}_i$  è il più corto vettore presente in  $\mathcal{L}$  per cui  $(\mathbf{b}_1, \dots, \mathbf{b}_i)$  può essere opportunamente esteso ad una base di  $\mathcal{L}$ .

Si può dimostrare che ogni Reticolo  $\mathcal{L}$  ammette una base ridotta secondo Minkowski: il problema di determinare tale base, a partire da una base qualunque di  $\mathcal{L}$ , è detto *Problema della Riduzione della Base di un Reticolo* (alla forma ridotta secondo Minkowski).

Dal fatto che una base  $B$ , ridotta secondo Minkowski, contiene il vettore più corto del Reticolo generato da  $\mathcal{L}$ , segue che tale problema è almeno tanto difficile quanto determinare il vettore non nullo di norma minima in un Reticolo: nel caso di alcune norme, si può provare che tale problema è intrattabile, mentre per altre norme (fra le quali la norma Euclidea), l'intrattabilità è solo una congettura. Pertanto, anche il Problema della Riduzione della Base di un Reticolo è ritenuto difficile, ed infatti nella pratica si utilizzano degli algoritmi approssimati per calcolare efficientemente delle basi che non si discostino eccessivamente dalla vera base ridotta.

### A.3 L'algoritmo LLL

Trattando della Crittografia a Chiave Pubblica, si ha spesso a che fare con la risoluzione di problemi di Matematica Discreta, come ad esempio la ricerca di soluzioni intere per una o più equazioni. Tali problemi possono essere ricondotti a questioni di *Teoria dei Numeri*, costruendo un'opportuno Reticolo la cui geometria rispecchi le caratteristiche del problema originale.

Fra i problemi noti in tale ambito, si considera adesso il *Problema della Approssimazione in un Reticolo*, vale a dire la determinazione, fra i punti di un dato Reticolo, di quello a distanza minima da un punto assegnato nello spazio Euclideo  $\mathbb{R}^n$ . Con un opportuno cambiamento del sistema di riferimento, tale problema si può riformulare come la ricerca di un punto del Reticolo (diverso dall'origine) la cui distanza dall'origine sia minima, ovvero avente minima norma non nulla. La risoluzione esatta di questo problema necessita di un tempo esponenziale; tuttavia sono note delle soluzioni efficienti per determinare un vettore la cui norma non si discosti troppo dal minimo effettivo.

Una di tali soluzioni è l'algoritmo LLL [LLL82]. Si tratta, in realtà, di un algoritmo approssimato per la riduzione di una base  $B = (b_1, \dots, b_n)$  alla sua forma ridotta secondo Minkowski  $M = (m_1, \dots, m_n)$ .

In breve, l'algoritmo LLL alterna un passo di riduzione della grandezza dei vettori di  $B$  ad un passo di riordinamento degli elementi della base, fino ad ottenere una nuova base  $B^*$ , detta LLL-ridotta, per cui risulta che:

$$\|b_i^*\| \leq 2^{k/2} \|m_i\|, \quad \forall i, 1 \leq i \leq n$$

Tale base  $B^*$  viene ottenuta eseguendo  $\mathcal{O}(n^6(\lg C)^3)$  operazioni aritmetiche elementari, dove  $C \doteq \sqrt{\|b_1\|^2 + \dots + \|b_n\|^2}$ .

Nella pratica, l'algoritmo LLL determina una base  $B^*$  i cui vettori approssimano molto meglio la base  $M$  di quanto non sia garantito dal risultato sopra riportato; inoltre, nel caso particolare del vettore  $b_1^*$ , ovvero quello che approssima il vettore del Reticolo di minima norma non nulla, è possibile dimostrare [Bab86] che, anche nel caso peggiore, la suddetta maggiorazione esponenziale può essere riscritta come:

$$\|b_1^*\| \leq 2^{k/4.6} \|m_1\|$$

## A.4 Un Problema di Programmazione Intera

Un'altra interessante questione che riguarda i Reticoli è *il Problema della Esistenza di Soluzioni Intere per un Sistema di Disequazioni Lineari*. Tale problema risulta di interesse crittografico, perché la potenza espressiva della Programmazione Intera gli conferisce una notevole flessibilità.

Si consideri un sistema di disequazioni lineari, espresso in forma matriciale, come  $A\mathbf{x} \leq \mathbf{b}$ . Tale sistema individua un poliedro  $\mathbf{K}$  in  $\mathbb{R}^n$ :

$$\mathbf{K} = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b}\}$$

Il problema dell'esistenza di una soluzione intera consiste nello stabilire se all'interno di  $\mathbf{K}$  vi sono o meno punti (diversi dall'origine) appartenenti al Reticolo  $\mathbb{Z}^n$  generato dalla base canonica.

In [Len83] è mostrato che tale problema può essere risolto in tempo polinomiale in  $n$ , seguendo un approccio *Divide and Conquer*. Il poliedro  $\mathbf{K}$  viene prima ricondotto ad una forma quasi-sferica, mediante un'opportuna trasformazione  $\tau$ , che causa il cambiamento della base del Reticolo  $\mathbb{Z}^n$ . Se i vettori della nuova base sono ancora corti e quasi-ortogonali, è possibile decidere il problema risolvendo al più  $n$  nuove istanze con una dimensione in meno, ottenute proiettando l'intera costruzione su un certo numero di iperpiani paralleli.

# Bibliografia

- [ACGS88] W. Alexi, B. Chor, O. Goldreich, and C. Schnorr, *RSA and Rabin Functions: Certain Parts are as Hard as the Whole*, SIAM Journal on Computing **17** (1988), no. 2, 262–280.
- [Bab86] L. Babai, *On Lovasz' Lattice Reduction and the Nearest Lattice Point Problem*, Combinatorica **6** (1986), 1–13.
- [BV96] D. Boneh and R. Venkatesan, *Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes*, Advances in Cryptology - Crypto '96 (Berlin) (N. Koblitz, ed.), Springer-Verlag, 1996, Lecture Notes in Computer Science Volume 1109, pp. 129–142.
- [DH76] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory **22** (1976), 644–654.

- [Fei73] H. Feistel, *Cryptography and Computer Privacy*, Scientific American (1973).
- [FHK<sup>+</sup>88] A. Frieze, J. Hastad, R. Kannan, J. Lagarias, and A. Shamir, *Reconstructing Truncated Integer Variables Satisfying Linear Congruences*, SIAM Journal on Computing **17** (1988), no. 2, 194–209.
- [GL89] O. Goldreich and L. Levin, *Hardcore Bits Based on any One-Way Function*, STOC '94, 1989.
- [Len83] H. Lenstra, *Integer Programming in a Fixed Number of Variables*, Mathematics of Operations Research **17** (1983), 538–548.
- [LLL82] A. Lenstra, H. Lenstra, and L. Lovasz, *Factoring Polynomials with Integer Coefficients*, Mathematische Annalen **261** (1982), 513–534.
- [Mau94] U. Maurer, *Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms*, Advances in Cryptology - Crypto '94 (Berlin) (Y. Desmedt, ed.), Springer-Verlag, 1994, Lecture Notes in Computer Science Volume 839, pp. 271–281.
- [MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1996.

- [RSA78] R. Rivest, A. Shamir, and L. Adelman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM (1978).
- [Sha49] C. Shannon, *Communication Theory of Secrecy Systems*, Bell Systems Technical Journal (1949), no. 4.
- [Sho97] V. Shoup, *Lower Bounds for Discrete Logarithms and Related Problems*, Advances in Cryptology - EuroCrypt '97 (Berlin) (W. Fumy, ed.), Springer-Verlag, 1997, Lecture Notes in Computer Science Volume 1233, pp. 256–266.
- [Sti95] D. Stinson, *Cryptography: Theory and Practice*, CRC Press, Boca Raton, FL, 1995.
- [VS00] G. Vasco and I. Shparlinski, *On the Security of Diffie-Hellman Bits*, Tech. Report TR00-045, ECCO, Luglio 2000.