Alma Mater Studiorum - Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Triennale in Informatica

IMPLEMENTAZIONE DI TEST DI CASUALITA' IN CRYPTOOL

Tesi di Laurea in Sicurezza e Crittografia

Relatore:
Ill.mo Dott. Ugo Dal Lago

Presentata da:
Carla Bressan

III Sessione
Anno Accademico 2011/2012

Introduzione

Premessa

In questa tesi mi sono proposta l'obiettivo di realizzare l'implementazione di alcuni semplici test di casualità per l'analisi della bontà di generatori pseudocasuali di numeri. La scelta di sviluppare la mia attività di tesi e di tirocinio nell'ambito della crittografia, è dovuta ad un mio interesse personale verso questo argomento nato durante un corso di perfezionamento in Didattica della Matematica che si è tenuto in questa Università nell'a.a. 1995-1996. In questo corso, nel ciclo di lezioni di Algebra, furono approfonditi alcuni concetti come l'algebra modulare, la fattorizzazione in numeri primi, la funzione φ di Eulero e altri che sono alla base della crittografia classica e moderna. Essendo il corso rivolto ai futuri insegnanti di Matematica della scuola dell'obbligo e dei primi due anni degli istituti superiori, tutti i temi erano stati scelti per i possibili utilizzi che avrebbero potuto avere nella prassi didattica. Ciò che di questa parte mi colpì particolarmente fu scoprire che alla base della crittografia vi erano concetti di Matematica fondamentalmente piuttosto semplici: divisibilità, numeri primi, fattorizzazione, algoritmo euclideo della divisione, tutte nozioni che sono presentate fin dal primo anno della scuola media. Da allora ho cominciato a leggere articoli di riviste e qualche libro per lo più divulgativo e a partecipare ad iniziative sull'argomento. Per esempio nell'anno 2004 nel contesto del progetto europeo "Mathematics in Europe", nell'ambito del Contratto Europeo: Diffusion and improvement of mathematical knowledge in Europe nel quale l'Università di Bologna parte-

cipava come partner, ho assistito al ciclo di conferenze *Matematica, Arte, Scienza e Tecnologia*, alcune delle quali erano proprio dedicate alla Sicurezza in Internet e alla Crittografia (quelle per esempio tenute dal Prof. Theo Mora dell'Università di Genova e quella del Prof. Viganò dell'Università di Verona) e alla rassegna *Matematica e Cinema II*, durante la quale fu proposto il film "Enigma" (Das Geheimnis, GB, Germany, USA, 2001) di Michael Apted. Attualmente insegno Matematica in una scuola media di Bologna, ma ho insegnato anche diversi anni in alcuni Licei di Bologna e il mio interesse personale negli anni si è tradotto anche in una esigenza lavorativa: trovandomi con dispiacere di fronte alla convinzione di molti studenti che la Matematica sia una disciplina distante dalla vita reale, inutile, troppo teorica e sempre uguale, ho spesso fatto appello alle mie modeste conoscenze di crittografia (soprattutto durante i primi anni di insegnamento nei licei), per dimostrare invece quanto la Matematica sia utilizzata quotidianamente da tutti e sia una disciplina attuale, moderna vivace e in continua evoluzione. La necessità di rendere più accattivanti alcune lezioni e di trovare nuovi spunti per renderle interessanti unita al mio personale interesse, hanno però anche evidenziato che le mie conoscenze, tutte da autodidatta, erano comunque troppo limitate. Avevo perciò deciso di approfondire questa parte durante il corso di Laurea in Informatica. Purtroppo, non mi è stato possibile seguire le lezioni del corso di Sicurezza per motivi di lavoro poiché già insegnavo quando ho cominciato questo ciclo corso di laurea. Ho quindi pensato di dedicare all'approfondimento di questo argomento il periodo della tesi e del tirocinio.

Il Percorso Concettuale della Tesi

La tesi è stata strutturata seguendo un percorso di apprendimento. Il primo capitolo è dedicato alla storia della crittografia classica, in un arco temporale che va dall'antichità alla Seconda Guerra Mondiale. Si è ritenuto importante inserire questa parte che potrebbe sembrare lontana dall'oggetto fondamentale della tesi perché si ritiene che la conoscenza di come si è giun-

ti a determinate scoperte presenti un alto valore formativo in quanto rende evidente la provvisorietà dei modelli scientifici che l'uomo si è costruito nel tempo e le intersezioni da sempre intercorse fra la scienza e gli altri campi del sapere. Questo capitolo ha comunque un carattere puramente introduttivo: sono stati scelti come momenti significativi tutti quelli che hanno costituito tappe fondamentali per quanto approfondito successivamente. Per esempio, ci si è soffermati sul cifrario di Cesare, che nel Capitolo 6 verrà utilizzato in una possibile esperienza didattica, e su quello di Vigenere che è stato un tentativo di superare le criticità del cifrario di Cesare. Da questi semplici cifrari si è passati a descrivere le macchine cifranti che costituiscono i primi esempi di automatizzazione del processo di cifratura e che hanno dato lo spunto per ricordare l'opera di Alan Turing e la sua influenza sullo sviluppo dell'informatica. Nel Capitolo 2 sono stati introdotti i principi fondamentali della crittografia moderna e la terminologia che verrà utilizzata nei capitoli successivi. In questo capitolo si comincia a delineare il ruolo della Matematica e dei suoi metodi formali come base della crittografia moderna. Nel terzo capitolo viene presentato il cifrario one-time pad e dimostrato che è un cifrario perfetto. In questo capitolo viene utilizzato il formalismo matematico e presentato il teorema di Shannon. Questo capitolo ha un ruolo chiave perché da un lato si mostra la possibilità teorica del cifrario one-time pad che si dimostra essere perfetto e dall'altro si mostra l'impossibilità pratica dell'utilizzo di questo cifrario. Le criticità che sono descritte brevemente in questo capitolo sono esposte con maggiori argomentazioni nel Capitolo 4 dove viene mostrato come poter superare l'idea di cifrari *matematicamente* sicuri a favore di schemi di cifratura *praticamente* sicuri. Viene formalizzato il concetto di schema di cifratura a chiave privata e introdotta la nozione di indistinguibilità come sinonimo di sicurezza. In questo capitolo viene presentato poi il concetto cruciale di pseudocasualità: come la sicurezza computazionale è un indebolimento della sicurezza perfetta, così la pseudocasualità è un indebolimento della pura casualità. La costruzione matematica presentata nel precedente capitolo che assicura l'esistenza di cifrari perfetti, mostra qui

tutta la sua debolezza. La teoria deve fare i conti con la pratica: *la chiave utilizzata nello schema di cifratura one-time pad non può essere casuale, ma deve essere pseudocasuale.* Questa limitazione pratica non pregiudica però il valore della costruzione teorica: infatti il risultato fondamentale di questo capitolo è che uno schema di cifratura così costruito risulta indistinguibile e quindi (per quanto dimostrato nel capitolo 3) *computazionalmente sicuro.* Giunti a questo punto, è necessario definire dei criteri per stabilire quando una stringa di bit che può essere utilizzata come seme per la generazione di chiavi da utilizzare nello schema precedentemente costruito, può essere considerata casuale. Il National Institute of Standards and Technology nella Special Pubblication 800-22rev1a (Aprile 2010) intitolata *A Statistical Test Suite for Random and Pseudorandom Number Generators* fornisce uno standard di validazione e una suite con una serie di test statistici per la prova e validazione di generatori di numeri pseudocasuali. Nel capitolo 5 viene appunto descritta la suite e illustrata la parte matematica che sta alla base dei test. Il codice dei test è messo a disposizione dal NIST in linguaggio C. Nel Capitolo 6 potrei dire che il cerchio si chiude riprendendo da dove iniziato e cioè dalla possibilità di utilizzare i concetti così attuali come la crittografia e la sicurezza nella pratica didattica. Viene presentato un portale di apprendimento della crittografia sviluppato da alcune Università e centri di ricerca di Germania e Austria. Il portale Cryptool è open source scritto in linguaggio C#, è tutt'ora in evoluzione e al suo interno contiene cinque progetti che sviluppano e integrano l'idea iniziale: creare un portale di e-learning nell'ambito della crittografia e della crittoanalisi. Sono stati anche implementati i primi quattro test di casualità proposti nella suite del NIST all'interno del progetto CrypTool2 che fornisce una interfaccia grafica innovativa che può essere utilizzata efficacemente nella didattica. Il codice fornito dal NIST è stato tradotto nel linguaggio C# inserendo i test come plugin nel progetto CrypTool2. Sono state effettuate alcune prove e presentati i risultati.

Osservazioni

L'attività svolta è stata molto interessante. Sono stati approfonditi concetti che sono assolutamente attuali e che comunemente utilizziamo tutti, anche se in modo inconsapevole. Le difficoltà incontrate sono state diverse: la scelta di quali momenti della storia della scienza sono stati significativi al fine del lavoro, il cimentarsi con concetti difficili senza aver seguito un ciclo di lezioni specifico. Vi sono poi state le difficoltà legate alla implementazione degli algoritmi per i test di casualità: anche se molto codice era fornito dal NIST, è stato necessario riscriverlo in un altro linguaggio e soprattutto adattarlo per poterlo inserire in un progetto collaborativo. A questo riguardo occorre dire che la documentazione è piuttosto scarna: pochissime spiegazioni sono fornite sull'utilizzo del portale e poche si trovano su come aggiungere nuove funzionalità. Per quanto riguarda poi l'idea di svolgere una lavoro che fosse riutilizzabile nell'insegnamento, occorre dire che una attività di questo genere opportunamente guidata e semplificata potrebbe essere efficacemente svolta nell'ultimo anno di un Liceo. Per completezza si è però ritenuto opportuno concludere il Capitolo 6 presentando anche una breve attività di laboratorio per la scuola media basata sul cifrario di Cesare e sull'utilizzo del portale che potrebbe invece essere svolta in una scuola media. Si tratta di una attività semplice, tuttavia i concetti di Matematica utilizzati sono significativi e può essere quindi utilizzata proprio come un esempio di Matematica applicata. Il portale CrypTool contiene poi numerosi strumenti e funzionalità che possono certamente rendere più accattivanti le spiegazioni ed essere validi spunti per successive attività nell'ambito della Matematica e fornire collegamenti con altre discipline (storia, inglese, italiano) che sono senza dubbio utili nel processo di apprendimento di ogni studente.

Indice

Capitolo 1

Introduzione Storica

Questo capitolo è dedicato alla storia della crittografia classica, in un arco temporale che va dall'antichità alla Seconda Guerra Mondiale. Non si tratta che di qualche accenno: vengono ricordate le tappe della storia della crittografia che sono state fondamentali per le applicazioni che vengono richiamate successivamente.

1.1 Un pò di Terminologia

La crittografia (dal greco *Kryptòs* che significa nascosto e *grpàhein* che significa scrivere), è l'insieme delle tecniche utilizzate per nascondere un messaggio scritto in modo che non risulti comprensibile ad altri che non siano il mittente (di seguito abbreviato con MITT) e il destinatario (DEST). Un tale messaggio viene definito crittogramma. L'operazione tramite la quale si nascondono le informazioni è chiamata cifratura (oppure crittazione), ed è effettuata tramite un apposito algoritmo chiamato cifrario; l'informazione o il messaggio da cifrare è noto come testo chiaro (plaintext) mentre il testo ottenuto dalla cifratura si dice testo cifrato o crittogramma (ciphertext). Con il termine decrittazione si intende la conversione da testo cifrato a testo chiaro. Se supponiamo che la cifratura e la decrittazione avvengono per mezzo di

qualche informazione che mittente e destinatario condividono a priori chiamata *chiave* si parla di crittografia a chiave privata (o a chiave simmetrica). In questo scenario la chiave condivisa serve a distinguere le parti che comunicano da ogni altro che possa intercettare la comunicazione (che si assume aver luogo attraverso un canale pubblico). Con il termine crittoanalisi sta ad indicare lo studio delle tecniche per decifrare i crittogrammi. Lo studio della crittografia e della crittoanalisi si chiama comunemente crittologia.

1.2 Crittografia Antica

La necessità di nascondere i messaggi si può dire antica quanto l'uomo: i primi tentativi di comunicazione segreta consistevano nel nascondere l'esistenza del messaggio, tecnica nota come *steganografia* pratica che fu utilizzata per esempio dai Persiani e dai Greci e dai Romani. Erodoto infatti racconta di un nobile persiano che fece tagliare a zero i capelli di uno schiavo fidato per potergli tatuare un messaggio sul cranio e una volta che i capelli furono ricresciuti lo mandò a destinazione con la sola istruzione di tagliarseli nuovamente. I Greci usavano incidere il testo del messaggio segreto su tavolette di legno e successivamente lo occultavano ricoprendolo di cera e infine scrivevano sulla cera un messaggio generico, magari una poesia d'amore per non destare sospetti; per leggere il vero messaggio il destinatario doveva grattare via la cera dalla tavoletta. Gli antichi Romani, invece, usavano spesso scrivere tra le righe di una generica lettera usando come inchiostro il succo di limone (ma anche aceto o latte) che, una volta asciutto, diviene invisibile, ma se si pone la lettera in prossimità di una fiamma il succo di limone si brucia più rapidamente della carta e fa apparire la scritta. Altre volte usavano il componimento acrostico (dal greco akròstichon - ciò che si trova all'inizio del verso), versi poetici nei quali l'unione delle lettere iniziali di ciascun verso forma una parola o una frase. La steganografia pone come fattore di sicurezza il fatto di non conoscere l'esistenza stessa dell'informazione. Nascondendo il messaggio segreto in maniera tale che nessuno possa trovarlo, funziona finché

nessuno si aspetta che qualcuno ne stia facendo uso. Lo svantaggio di questo metodo è però la totale perdita di segretezza nel momento in cui il messaggio viene intercettato.

Parallelamente alla steganografia si sviluppa la crittografia che invece mira a nascondere il testo del messaggio e non la sua esistenza. A questo scopo si altera testo del messaggio per mezzo di un procedimento concordato tra mittente e destinatario in modo che se dovesse essere intercettato da qualcuno che non sia il destinatario, questo sia comunque inutilizzabile.

I primissimi esempi di crittografia sono stati scoperti in alcuni geroglifici egiziani risalenti a più di 4500 anni fa. Dagli scritti di Plutarco si sa che gli Spartani utilizzavano un rudimentale sistema crittografico per le comunicazioni di guerra che consisteva in un piccolo bastone di legno (la *scitala*) con una striscia di pelle arrotolata su cui veniva scritto il messaggio segreto. Una volta srotolata la striscia di pelle era impossibile decifrare il messaggio. Il popolo ebraico ideò invece il codice di Atbash, utilizzato nel libro di Geremia per cifrare il nome della città di Babilonia. Svetonio nella Vita dei dodici Cesari, un'opera del II secolo d.C., racconta che Giulio Cesare usava per le sue corrispondenze riservate un metodo di cifratura molto semplice, nel quale ogni lettera del testo veniva sostituita dalla lettera che la segue di tre posti nell'alfabeto. Più in generale si dice codice di Cesare un codice nel quale la lettera del messaggio chiaro viene spostata di un numero fisso di posti, non necessariamente tre. Sono possibili 26 codici di Cesare diversi poiché l'alfabeto internazionale è composto da 26 caratteri. Questo tipo di cifratura è detta sostituzione monoalfabetica. Il codice di Cesare venne violato grazie al contributo dato dalla scoperta del matematico arabo Al-Kindi (XI secolo circa) che in ogni lingua la frequenza di uso di ogni lettera è fissa; questo è vero in modo rigoroso solo per testi lunghi ma spesso testi anche corti hanno frequenze non molto diverse da quelle previste. Confrontando la frequenza delle lettere nel testo cifrato con la frequenza nota delle lettere di una lingua è possibile conoscere la tabella di corrispondenza delle lettere del codice.

Nel 1586 il diplomatico e crittografo francese Blaise de Vigenè pubblicò

uno dei più semplici cifrari polialfabetici, considerato per secoli inattaccabile. La forza della cifratura di Vigenère sta nell'utilizzare non uno, ma bensì 26 alfabeti cifranti per cifrare un solo messaggio. Il primo passo consiste infatti nella stesura della tavola di Vigenère: si tratta di un normale alfabeto chiaro

Chiaro	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
26	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Tabella 1.1 Tavola di Vigènere

di 26 lettere seguito da 26 alfabeti cifranti, ciascuno spostato a sinistra di una lettera rispetto al precedente. Perciò, la riga numero 1 rappresenta un alfabeto cifrante con uno spostamento di Cesare pari a 1, la riga 2 rappresenta un alfabeto cifrante con uno spostamento di Cesare pari a 2, e così via. La fila in cima al quadrato, in caratteri minuscoli, è un alfabeto ordinario di 26 lettere, che permette di cifrare qualunque lettera del testo chiaro tramite uno dei 26 alfabeti sottostanti. Per esempio, usando l'alfabeto cifrante numero 2, A è cifrata come C, mentre usando l'alfabeto numero 12, è cifrata come M. La cifratura di Vigenère comporta che per ciascuna lettera del messaggio si usi una diversa riga della tavola (cioè un diverso alfabeto cifrante). Per esempio, il mittente potrebbe crittare la prima lettera in base alla riga 5,

la seconda in base alla riga 14, e così via. Perché il messaggio possa esser decifrato, è indispensabile che il destinatario sappia quale riga della tavola di Vigenère è stata usata per ciascuna lettera; in altri termini, la continua sostituzione degli alfabeti cifranti deve essere effettuata non a casaccio, ma in base a una regola precisa. Quest'ultima è riassunta da una parola o frase chiave. Il grande vantaggio della cifratura di Vigenère è la sua resistenza all'analisi delle frequenze. Inoltre la cifratura di Vigenère ammette un enorme numero di chiavi. Mittente e destinatario possono scegliere qualunque parola del dizionario, qualunque combinazione di parole, possono perfino ricorrere a neologismi. La via di controllare tutte le chiavi possibili è dunque impercorribile. Viste la sua forza e le difficoltà in cui si dibattevano le scritture segrete tradizionali, viene spontaneo pensare che il metodo dello studioso francese avesse avuto un enorme successo. Invece, la cifratura di Vigenère non suscitò alcun entusiasmo, e pur apparendo priva di punti deboli, fu pressoché ignorata per ben due secoli e fu ripresa come vedremo soltanto durante la Seconda Guerra Mondiale.

1.3 La Crittografia dalla Seconda Metà del XIX Secolo alla Grande Guerra

Fino alla prima metà del XIX secolo la corrispondenza era esclusivamente cartacea ed era recapitata dai servizi postali. Tra la seconda metà del XIX secolo e il XX secolo, l'invenzione del telegrafo, del telefono e della radio cambiarono radicalmente il modo di comunicare, rendendo possibile la trasmissione di messaggi in modo pressoché istantaneo anche da luoghi molto distanti. Questi nuovi mezzi di comunicazione, la radio in particolare, rendevano però ancora più facili e frequenti le intercettazioni da parte di nemici; il ricorso alla crittografia diventò, quindi, inevitabile, come la necessità di cifrari sempre più sofisticati.

Nel 1863 il colonnello prussiano Friedrich Kasiski pubblica il primo metodo di decrittazione del cifrario di Vigenère. Il suo metodo si basava sull'os-

servazione che porzioni ripetute di messaggio cifrate con la stessa porzione di chiave risultano segmenti di testo cifrato identici.

Allo scoppio della Prima Guerra Mondiale gli unici paesi organizzati con veri e propri "Uffici Cifra" in cui venivano decifrati i messaggi radio dei nemici erano Francia e Austria, mentre i Russi solo verso la fine della guerra cifravano i loro messaggi. In Italia la crittografia in questo periodo viene pressoché ignorata e si dovrà attendere l'entrata in Guerra nel 1915 per rendersi conto del ritardo accumulato in campo crittografico, e porvi rimedio. I crittografi britannici si riunivano nella Stanza 40, nome della stanza dell'ammiragliato inglese sede dell'ufficio crittografico preposto alla violazione dei codici cifrati tedeschi. Negli USA fu adoperato come Ufficio Cifra il reparto crittologico dei laboratori Riverbanks di Chicago, nel quale lavorava anche William Friedmann destinato a divenire il massimo crittologo e crittanalista degli Stati Uniti.

1.4 La Crittografia nella Seconda Guerra Mondiale

Nella Prima Guerra Mondiale si era imposta l'esigenza di comunicazioni sicure e di conseguenza anche dello studio di metodi per la decifrazione di messaggi segreti, ma fu nella Seconda Guerra mondiale che la crittografia assunse un ruolo di primaria importanza. La necessità di una progressiva automatizzazione dei metodi di cifratura e decifratura, portarono poi alla nascita delle *macchine cifranti*, apparati meccanici che resero i sistemi crittografici non tanto più sicuri, ma senza dubbio più veloci. L'idea di fondo di tutte le macchine cifranti è in genere quella di uno o più dischi o cilindri rotanti detti *rotori* che permettono di trasformare automaticamente le lettere del testo chiaro in quelle del testo cifrato e viceversa. Ogni rotore ha inciso sopra una permutazione dell'alfabeto a 26 lettere e gira attorno a un asse: questo permette di cifrare ogni lettera immessa con un alfabeto diverso. Le macchine cifranti possono quindi essere ritenute una versione meccanica del

cifrario di Vigenère. Il rotore rappresenta la caratteristica principale delle macchine cifranti ma, al contempo, anche il suo punto debole, in quanto, dopo 26 rotazioni, il disco torna nella posizione iniziale. Una macchina cifrante ad un rotore solo quindi ripete il suo schema di crittografia ogni 26 lettere. Aggiungendo altri rotori è possibile aumentare il numero di lettere prima che il testo venga cifrato con lo stesso schema. Possiamo perciò dire che la sicurezza aumenta esponenzialmente con l'aumentare del numero dei rotori. Una delle più celebri macchine cifranti che cominciarono a diffondersi nella prima metà del XX secolo è sicuramente Enigma, brevettata dall'ingegnere tedesco Arthur Scherbius nel 1918 e adottata dell'esercito e dalla Marina tedesca durante la Seconda Guerra Mondiale. La versione base del dispositivo era costituita da tre componenti collegati tra loro con fili elettrici:

- una tastiera per immettere le lettere del testo in chiaro;
- un'unità scambiatrice che cifra la lettera trasformandola nel corrispondente elemento del crittogramma;
- un visore con varie lampadine che illuminandosi indicano la lettera da inserire nel testo cifrato.

Lo scambiatore rappresentava la parte più importante della macchina: consisteva in uno spesso disco di gomma attraversato da una fitta rete di fili provenienti dalla tastiera. Questi fili entravano nello scambiatore e dopo un percorso formato da vari gomiti emergevano dalla parte opposta. Lo schema interno dello scambiatore determinava in pratica un alfabeto cifrante utilizzabile per una semplice cifratura a sostituzione monoalfabetica. Così com'è il meccanismo presentava ancora il problema della ripetizione che è comunemente sinonimo di cifratura debole. Per superarlo vennero introdotti un secondo e un terzo scambiatore. Il secondo compiva una rotazione parziale soltanto dopo che il primo aveva compiuto un intero giro e allo stesso modo faceva il terzo, basandosi sul secondo. In questo modo la macchina di Scherbius poteva disporre di 26 x 26 x 26 = 17576 procedure di sostituzione diverse. La macchina venne poi arricchita ulteriormente con un pannello a

prese multiple che permetteva di scambiare coppie di lettere all'inizio della cifratura e un anello che regolava i tempi di rotazione dei rotori. Tutto questo rendeva impossibile la crittoanalisi tramite i metodi tradizionali di analisi e fece sì che Enigma potesse permettere un enorme numero di chiavi, ovvero di configurazioni iniziali della macchina. L'unico modo per decifrare il crittogramma era quindi quello di possedere una macchina Enigma configurata esattamente come quella con cui si era cifrato il messaggio e batterne le lettere sulla tastiera. La configurazione delle macchine tedesche veniva cambiata ogni 24 ore seguendo un determinato protocollo da tenere in totale segretezza in quanto, se gli alleati ne fossero venuti in possesso, avrebbero potuto decifrare facilmente ogni messaggio.

Grazie all'utilizzo di Enigma il metodo crittografico tedesco appariva insormontabile. Solo i Polacchi, che intuivano le mire espansionistiche della Germania ai loro danni, non si diedero per vinti. Molti anni dopo la fine della guerra si seppe che, in effetti, già nel 1932 l'Ufficio Cifra polacco, guidato dal matematico Rejewski, era riuscito a trovare il modo di forzare la macchina Enigma. Nell'agosto del 1939 i Britannici costituirono la scuola dei codici e dei cifrari a Bletchley Park, dove reclutarono i migliori crittoanalisti, matematici e scienziati, tra i quali ricordiamo Alan Turing che ideò una serie di tecniche per violare i cifrari tedeschi, incluso il metodo di Bomba, una macchina elettromeccanica in grado di decodificare codici creati mediante la macchina Enigma. Turing morì suicida a soli 41 anni, ma il suo lavoro ebbe vasta influenza sullo sviluppo dell'informatica, grazie alla sua formalizzazione dei concetti di algoritmo e calcolo mediante la cosiddetta macchina di Turing, che a sua volta ha svolto un ruolo significativo nella creazione del moderno computer. Per questi contributi Turing è solitamente considerato il padre della scienza informatica e dell'intelligenza artificiale.

Sfruttando anche le conoscenze raggiunte dagli alleati polacchi, durante la guerra, gli inglesi continuarono a forzare sistematicamente i messaggi cifrati con Enigma e dal 1941 anche quelli cifrati con la più sofisticata macchina Lorenz.

La crittografia ebbe un ruolo di fondamentale importanza durante tutta la durata della guerra e, fu fondamentale anche per lo Sbarco in Normandia. Infatti Eisenhower e Montgomery erano in grado di leggere tutti i messaggi degli alti comandi tedeschi, che usavano la macchina Lorenz; ebbero così conferma che Hitler aveva creduto alla falsa notizia di un imminente sbarco alleato nei pressi di Calais, e aveva concentrato le sue migliori truppe in quella zona. Poterono quindi ordinare lo sbarco in Normandia sicuri che avrebbe incontrato ben poca resistenza. Fin dal 1940 inoltre, gli americani avevano realizzato Magic, una macchina in grado di decrittare i messaggi giapponesi cifrati. Questa consentì, ad esempio, agli americani di vincere la Battaglia delle Midway, conoscendo fin nei dettagli i piani dell'esercito nipponico. E' possibile che gli americani fossero già a conoscenza anche dell'attacco di Pearl Arbour e decisero di non impedirlo, forse per convincere l'opinione pubblica della necessità dell'entrata in guerra. Una teoria più prudente sostiene che gli Americani sapevano che il Giappone stava per attaccare, ma non sapevano dove. Certo è che al momento dell'attacco nella baia di Pearl Harbour non c'era nemmeno una portaerei e, in definitiva, furono affondate solo alcune navi vecchie e di importanza non fondamentale per la guerra. Alla fine della guerra il gen. Marshall ammise che in molti casi di importanza "non vitale" gli alleati dovettero fingere di non conoscere i messaggi cifrati nemici, anche al costo di perdite umane, tale era il timore che tedeschi e giapponesi si accorgessero che i loro cifrari venivano sistematicamente decrittati.

La storia degli schemi classici di cifratura è affascinante sia per i metodi usati, sia per l'influenza che la crittografia e la crittoanalisi hanno avuto nella storia umana (come nella Seconda Guerra Mondiale). Qui abbiamo provato a dare solo un assaggio di alcuni dei metodi basilari, focalizzandoci su ciò che la moderna crittografia può aver imparato da questi tentativi.

Capitolo 2

Crittografia Moderna

In questo capitolo verranno introdotti i principi fondamentali della crittografia moderna e la terminologia che verrà utilizzata nei capitoli successivi. Si comincia inoltre a delineare il ruolo della Matematica e dei suoi metodi formali come base della crittografia moderna.

2.1 Introduzione

Fino al XX secolo la crittografia può essere considerata una forma di arte. Costruire codici sicuri o forzarne di esistenti aveva più a che fare con la creatività e abilità personale di chi si cimentava in questa sfida che con la scienza. La base teorica dei metodi crittografici era molto limitata e non si aveva una nozione ben definita di cosa rendesse un sistema di codifica un "buon codice".

Verso la fine del XX secolo questa immagine della crittografia cambiò radicalmente portando ad uno studio rigoroso della crittografia come *scienza*. Inoltre la crittografia oggi comprende molto di più che lo scambio di messaggi segreti: riguarda ad esempio il problema dell'autenticazione dei messaggi, della firma digitale, dei protocolli per lo scambio di chiavi segrete, i protocolli di autenticazione, le aste elettroniche e le elezioni, le transazioni economiche

digitali e molto altro ancora. La moderna crittografia è lo studio scientifico delle tecniche per rendere sicure le informazioni digitali, le transazioni e i calcoli distribuiti.

Un'altra importante differenza con la crittografia classica è su chi la usa. Storicamente i principali utenti della crittografia erano i militari e le organizzazioni di sicurezza e spionaggio, ora le applicazioni della crittografia sono una parte integrante di ogni sistema computerizzato. Per esempio, ogni utente, magari inconsapevolmente, si affida alla crittografia ogni volta che accede ad un sito web sicuro. I metodi della crittografia sono poi usati per rafforzare il controllo di accesso nei sistemi operativi multi-utente e per prevenire la conoscenza da parte dei ladri di informazioni segrete presenti in computer rubati. Inoltre i metodi di protezione del software impiegano crittazione, autenticazione e altri strumenti per evitarne la copia.

Alla luce di queste considerazioni, possiamo dire con ragione che la crittografia da arte che riguardava le comunicazioni militari segrete è diventata una vera e propria scienza che aiuta a rendere sicuri i sistemi per le persone comuni di tutto il mondo e sta diventando un elemento sempre più centrale nella teoria dell'informazione.

2.2 I Principi Base della Moderna Crittografia

In questa parte si vuole sottolineare alcuni principi che distinguono la moderna crittografia da quella classica. Ne vediamo tre:

1. Principio 1: il primo passo nel risolvere qualunque problema crittografico è la formulazione di una precisa definizione di sicurezza;

2. Principio 2: quando la sicurezza di un sistema di crittografia si basa su ipotesi non dimostrate, queste ipotesi devono essere chiaramente dichiarate. Inoltre queste ipotesi devono essere il meno possibile;

3. Principio 3: le costruzioni crittografiche devono essere accompagnate da una rigorosa prova di sicurezza rispetto alle definizioni formulate in accordo col principio 1 e relativamente alle ipotesi dichiarate in accordo al principio 2.

Principio 1 – Formulazione di definizioni esatte
Le definizioni formali di sicurezza sono prerequisiti essenziali per la progettazione, l'uso e lo studio di ogni protocollo crittografico. Sono indispensabili per la progettazione perché se si ha chiaro cosa si vuole ottenere si possono rivolgere meglio gli sforzi e valutare in modo più corretto la qualità di ciò che si è costruito. Inoltre, per quanto riguarda l'uso degli schemi di cifratura, avere una definizione precisa della sicurezza fornita da un certo schema ci permette di sapere quale schema usare in un certo ambito. Per esempio, potremmo decidere di usare in un certo contesto uno schema meno sicuro in generale (anche se sufficiente nel caso in oggetto), ma più efficiente. Inoltre tali definizioni di sicurezza sono indispensabili anche perché permettono di confrontare due schemi di cifratura basandosi su criteri differenti che non la sola efficienza.

Principio 2 – Fiducia nelle ipotesi
La sicurezza di molti schemi di cifratura non si può provare incondizionatamente. Pertanto la sicurezza si basa su alcune ipotesi. Queste ipotesi devono essere dichiarate precisamente per tre ragioni principali:

1. Le ipotesi sono dichiarazioni non provate, ma considerate vere. E' necessario perciò che queste ipotesi siano studiate approfonditamente per essere sicuri che siano vere. Se queste ipotesi non sono dichiarate in modo chiaro è difficile poterle studiare;

2. Conoscere le ipotesi su cui due schemi di cifratura si basano permette di scegliere fra schemi ugualmente efficienti quello che ha le ipotesi più semplici o più ragionevoli nei diversi casi;

3. Una prova matematica può essere fornita solo di fronte a ipotesi precise.

Principio 3 – Rigorose prove di sicurezza
I primi due principi conducono naturalmente al terzo. Il fatto che debbano essere usate definizioni esatte e ipotesi precise implica che sia possibile una prova di sicurezza. Tale prova è necessaria perché i danni che possono derivare da una falla nello schema possono essere molto costosi. Inoltre la sicurezza di un protocollo crittografico non può essere provata come si fa solitamente con il software. Per esempio il fatto che un testo cifrato appaia incomprensibile e quindi che gli algoritmi di cifratura funzionino, non significa affatto non sia possibile forzare lo schema. D'altra parte, anche se in un certo software è presente un baco, generalmente esso produce lo stesso dei risultati attendibili perché l'errore potrebbe manifestarsi solo per particolari dati d'ingresso e rimanere quindi inosservato per molto tempo; o che il calcolatore stesso sia soggetto a sporadico malfunzionamento; o che il risultato sia interpretato scorrettamente. Inoltre non essendo gli utenti interessati a far si che il loro software fallisca, non utilizzano metodi complessi per trovarne gli errori. Al contrario chi vuole forzare un algoritmo di cifratura vuole assolutamente trovare i possibili errori per forzare lo schema e quindi fa tutti gli sforzi possibili in tale senso. Dunque anche un algoritmo «sbagliato» può essere tollerato, o addirittura preferito ad algoritmi corretti ma molto più lenti, se esso genera risultati errati con probabilità inferiore a quella che si verifichino errori per altre cause.

2.3 Crittografia a Chiave Privata

Come abbiamo già detto, nello scenario di crittografia a chiave privata due parti condividono qualche informazione segreta chiamata chiave e la usano quando vogliono comunicare in modo segreto l'un l'altra. La parte che trasmette il messaggio per prima (il mittente) usa la chiave per cifrare il messaggio prima di inviarlo e la parte che lo riceve (il destinatario) usa la

stessa chiave per decifrarlo. Per questo motivo la crittografia a chiave privata è detta anche *simmetrica*, al contrario di quella asimmetrica dove il mittente e il destinatario non condividono alcuna informazione segreta e chiavi differenti sono utilizzate per il processo di cifratura e per quello di decifrazione. In questo contesto, una ipotesi implicita è che le parti comunicanti abbiano un modo per condividere le chiavi in modo segreto. Ciò è abbastanza plausibile in ambito militare, perché mittente e destinatario potrebbero incontrarsi fisicamente in un posto sicuro per mettersi d'accordo sulla chiave da usare. In molti scenari moderni, le parti non possono incontrarsi fisicamente, ma lo stesso i metodi a chiave privata sono sufficienti e di largo uso; ad esempio per la cifratura del disco di un computer, dove lo stesso utente (in diversi istanti di tempo) usa una chiave segreta per leggere o scrivere sul disco. Inoltre la crittografia chiave privata è spesso usata assieme a metodi asimmetrici. Lo schema di crittografia a chiave privata è composto da tre algoritmi:

1. l'algoritmo di generazione della chiave Gen che è probabilistico e produce una chiave k scelta in base ad una qualche distribuzione tipica dello schema;

2. l'algoritmo di decrittazione Enc che prende in input una chiave k e un testo in chiaro m e produce un testo cifrato c. Indicheremo con $\text{Enc}_k(m)$ la cifratura del testo m usando la chiave k;

3. l'algoritmo di decifratura Dec che prende in input una chiave k e un testo cifrato c e produce in output un testo in chiaro m. Indicheremo la decifratura del testo cifrato c usando la chiave k con $textscDec_k(c)$.

L'insieme delle chiavi possibili è chiamato spazio delle chiavi e indicato con *Key*. Quasi sempre Gen semplicemente sceglie una chiave a caso nello spazio delle chiavi in modo uniforme. L'insieme dei messaggi che possono essere supportati dall'algoritmo di cifratura si indica con *Msg* e si chiama spazio dei messaggi. Poiché ogni testo cifrato è ottenuto dalla cifratura di qualche testo in chiaro attraverso la chiave k, gli insiemi *Key* e *Msg* assieme definiscono l'insieme di tutti i possibili testi cifrati che sarà indicato con

Critto. Uno schema di cifratura è quindi completamente determinato dai tre algoritmi (Gen, Enc, Dec) e dallo spazio dei messaggi *Msg.* Per essere corretto lo schema di cifratura dovrà essere tale che:

$$Dec_k(Enc_k(m)) = m$$

cioè decifrando il testo cifrato devo riottenere il testo in chiaro originale. Così come lo abbiamo definito, lo schema di cifratura funziona nel seguente modo:

a) Si esegue Gen per ottenere la chiave k;

b) La chiave k è condivisa tra le parti;

c) Il mittente calcola $c := \text{Enc}_k(m)$;

d) Il mittente invia c al destinatario attraverso un canale pubblico;

e) Il destinatario calcola $m := \text{Dec}_k(c)$;

Da questo schema di funzionamento, risulta chiaro che chiunque sia in possesso della chiave e dell'algoritmo Dec intercettando il messaggio sarebbe in grado di decifrarlo. Sembrerebbe quindi che per mantenere sicure le comunicazioni occorra mantenere segreti sia l'algoritmo di decifrazione Dec che la chiave k. Nel 1883, Augusto Kerckhoffs in un articolo apparso nel Journal des sciences militaires intitolato "La cryptographie militaire" sulle tecniche dell'utilizzo della crittografia nella strategia militare, mette in luce alcuni importanti principi per i cifrari militari. Uno dei più importanti di questi principi, noto come principio di Kerckhoffs è il seguente:

> *L'algoritmo di cifratura non deve essere tenuto segreto e può essere conosciuto dall'avversario senza che ciò provochi falle nella sicurezza.*

Con questo principio Kerckhoffs afferma che uno schema di cifratura deve essere progettato in modo che sia sicuro anche se un avversario conosce i dettagli di tutti gli algoritmi tranne la chiave. Soltanto la chiave quindi

costituisce l'informazione da tenere segreta per assicurare la sicurezza dell'intero sistema. Ci sono tre principali argomenti in favore del principio di Kerckhoffs. Il primo è che è molto più semplice per le parti scambiarsi in modo sicuro e mantenere nascosta una chiave che può essere una stringa di pochi bit piuttosto che un algoritmo che ha dimensioni centinaia di volte maggiori. Il secondo motivo è che nell'eventualità la chiave sia scoperta, è molto più semplice cambiare la chiave che non modificare l'algoritmo di cifratura, tanto che in effetti è una pratica di sicurezza sostituire spesso la chiave di cifratura anche se non è stata scoperta. Il terzo motivo è che nel caso di comunicazioni che coinvolgono molte persone è molto più semplice usare lo stesso algoritmo, ma chiavi differenti piuttosto che molti algoritmi e la stessa chiave.

Dalle osservazioni precedenti segue che l'inviolabilità di un protocollo dipende fortemente da un'accurata scelta delle chiavi segrete impiegate negli algoritmi di cifratura e decifrazione. Il primo requisito di una chiave è quello di non essere facilmente prevedibile da parte di un crittoanalista: per questo le chiavi devono essere generate con un procedimento casuale. È dunque alla possibilità di disporre di sequenze casuali, e al significato stesso del caso, che dobbiamo anzitutto rivolgerci.

Benché le sequenze casuali esistono e sono in larga maggioranza, non è semplice individuare sorgenti casuali, al punto che la loro stessa esistenza è al centro di un dibattito filosofico. In particolare è difficile garantire che la generazione di un bit avvenga in modo indipendente dalla generazione degli altri, poiché nella realtà ogni esperimento modifica l'ambiente circostante e può influenzare l'esperimento successivo. Nella pratica la perfetta casualità di una sorgente non potrà essere garantita e dovremo accettare alcuni compromessi. Per la generazione di sequenze *brevi* esistono due approcci che presentano sufficienti garanzie di casualità. Uno sfrutta la presunta aleatorietà di alcuni fenomeni fisici come l'intervallo di tempo tra l'emissione di particelle durante il decadimento di un materiale radioattivo, o i valori campionati del segnale proveniente da un microfono. L'altro impiega alcuni

processi software rilevando per esempio come valori casuali lo stato dell'orologio interno di un calcolatore o la posizione della testina su un disco rigido. I due approcci producono risultati ragionevolmente imprevedibili se non si ha accesso fisico ai dispositivi utilizzati, i quali altrimenti potrebbero essere persino manomessi falsando ad arte il processo di generazione. Il loro impiego è però problematico per difficoltà di realizzazione, e perché le sequenze generate si considerano tanto più casuali quanto più sono brevi. Torneremo in seguito sul problema della generazione della chiave e vedremo come in genere si ricorre a un diverso meccanismo algoritmico che ricerca la casualità all'interno di alcuni processi matematici.

Oggi il principio di Kerckhoffs è portato alle sue estreme conseguenze, tanto che non solo la sicurezza non si affida sulla segretezza dell'algoritmo, ma addirittura l'algoritmo deve essere pubblico. I vantaggi di questo approccio sono molteplici e includono i seguenti:

a) Gli algoritmi che sono resi pubblici sono studiati e sottoposti al giudizio di molti esperti e pertanto possono essere scoperte falle in essi e resi più sicuri.

b) E' meglio che le eventuali falle di sicurezza siano scoperte da esperti che le cercano per migliorare l'algoritmo piuttosto che da potenziali nemici.

c) Se la sicurezza del sistema si basasse sulla segretezza dell'algoritmo, allora pratiche di reverse engineering del codice potrebbero minacciare l'integrità.

d) Progetti di cifratura pubblici permettono la standardizzazione delle procedure.

L'attacco a un sistema crittografico ha l'obiettivo di forzare il sistema, ma il metodo scelto e il suo livello di pericolosità dipendono dalle informazioni in possesso del crittoanalista. I tipi di attacco principali in ordine di gravità sono: Ciphertext-only attack: L'avversario ha a sua disposizione i testi cifrati

di un insieme di messaggi, tutti codificati con lo stesso algoritmo di cifratura. Lo scopo del crittoanalista è quello di recuperare il plaintext della maggior parte dei messaggi cifrati oppure la chiave di codifica. Known-plaintext attack: L'avversario è a conoscenza di una serie di coppie $(m_1, c_1), ..., (m_r, c_r)$ e vuole determinare il testo in chiaro di qualche altro c_h (di cui non conosce il corrispondente testo in chiaro). Chosen-plaitext attack: L'avversario si è procurato una serie di coppie $(m_1, c_1), ..., (m_r, c_r)$ relativamente a messaggi in chiaro che lui ha opportunamente scelto in modo tale da recuperare il maggior numero di informazioni per recuperare la chiave oppure un algoritmo per decifrare qualsiasi messaggi cifrato con la stessa chiave; Chosen-ciphertext attack: L'avversario si è procurato una serie di coppie $(m_1, c_1), ..., (m_r, c_r)$ relativamente a testi cifrati che lui ha opportunamente scelto. Lo scopo dell'avversario è ancora di determinare il testo in chiaro che è stato cifrato e la cui decifratura non è in grado di ottenere direttamente. I primi due tipi di attacco sono passivi, nel senso che l'avversario riceve alcuni testi cifrati (e forse alcuni testi in chiaro corrispondenti) e quindi lancia il suo attacco. Gli ultimi due sono attivi perché l'avversario può richiedere la cifratura e la decifratura a sua scelta . I primi due attacchi sono realistici: nel ciphertext-only attack l'avversario ha bisogno solo di intercettare la comunicazione che avviene nel canale pubblico mentre nel known-plaintext attack si suppone che l'avversario in qualche modo possa ottenere il testo in chiaro corrispondente al testo cifrato che sta osservando. Anche questo è ragionevolmente realistico. Per esempio la cifratura potrebbe essere utilizzata per tenere segreti gli utili trimestrali di un'azienda fino alla loro data di uscita, sicché chiunque intercettasse il testo cifrato successivamente avrebbe anche il testo in chiaro.

Differenti applicazioni della cifratura possono richiedere che lo schema di cifratura resista a differenti tipi di attacco e potrebbe essere preferibile scegliere uno schema più debole, ma sufficiente per le necessità contingenti, piuttosto che uno più resistente, ma meno efficiente.

Capitolo 3

Cifrari Perfetti

Nel seguito viene illustrato il cifrario one-time pad e la dimostrazione che si tratta di un cifrario perfetto e presentato il teorema di Shannon in cui si afferma l'inutilità pratica dei cifrari perfetti.

3.1 Definizioni e Proprietà Principali

Ricordiamo la sintassi che è stata introdotta nel precedente capitolo. Uno schema di cifratura è definito da tre algoritmi Gen, Enc, e Dec, da uno spazio dei messaggi Msg (con $|Msg| > 1$). L'algoritmo di cifratura Gen è probabilistico e fornisce una chiave k. Sia Key lo spazio delle chiavi, che supponiamo infinito. L'algoritmo Enc prende in input una chiave $k \in Key$ e un messaggio $m \in Msg$ e fornisce in output un testo cifrato che indichiamo con $\text{Enc}_k(m)$. L'algoritmo Enc può essere probabilistico e quindi può fornire differenti testi cifrati su input diversi volte. Sia *Critto* l'insieme dei possibili testi cifrati che possono essere forniti da $\text{Enc}_k(m)$ per ogni possibile scelta di k e m. L'algoritmo Dec prende in input una chiave k e un testo cifrato c e fornice un messaggio in chiaro m. Supponiamo che Dec sia deterministico. In un famoso articolo del 1949 Claude Shannon formalizzò il processo crittografico mediante un modello matematico. Fino a quel momento uno schema

di cifratura era informalmente considerato perfetto se la sua sicurezza era garantita qualunque fosse l'informazione carpita sul canale di trasmissione: in un tale schema il crittoanalista, esaminando un crittogramma c, non doveva poter acquisire sul messaggio m alcuna conoscenza di cui non disponesse già da prima. Il contributo cruciale di Shannon fu la formalizzazione matematica del concetto di *conoscenza* legata ad una comunicazione segreta, e la conseguente definizione formale di cifrario perfetto. Seguiamo dunque la sua impostazione. La comunicazione tra un mittente MITT e un destinatario DEST è definita come un processo stocastico in cui il comportamento del mittente è descritto da una variabile aleatoria M che assume valori nello spazio Msg dei messaggi, e le comunicazioni sul canale sono descritte da una variabile aleatoria C che assume valori nello spazio $Critto$ dei crittogrammi. La distribuzione di probabilità della variabile M dipende dalle caratteristiche della sorgente, cioè dalla propensione del mittente a spedire diversi messaggi. Per ogni $m \in Msg$ e per ogni $c \in Critto$ definiamo:

- $P(M = m)$ come la probabilità che il mittente voglia spedire il messaggio m al destinatario;
- $P(M = m|C = c)$ come la probabilità a posteriori che il messaggio inviato sia effettivamente m dato che c è il crittogramma in transito.

Per accertare l'assoluta sicurezza del sistema crittografico poniamoci nella situazione di massimo pessimismo in cui l'avversario che ha intercettato c sia in possesso di tutta l'informazione possibile sul sistema tranne la chiave segreta. In particolare egli conosce la distribuzione di probabilità con cui il mittente genera i messaggi, il cifrario utilizzato e lo spazio Key delle chiavi. Possiamo ora porre la definizione di Shannon:

Definizione 3.1. : Uno schema crittografico (GEN, ENC, DEC) è perfetto se, per ogni distribuzione di probabilità su Msg, per ogni messaggio $m \in Msg$ e per ogni $c \in Critto$ per cui vale $P(C = c) > 0$ risulta

$$P(M = m) = P(M = m|C = c)$$

A parole, un cifrario è perfetto se la probabilità che il mittente invii un messaggio m è uguale alla probabilità che il messaggio inviato sia effettivamente m se il crittogramma c transita sul canale, e questa proprietà vale per qualunque m e c. Quindi fissato un particolare messaggio m, la probabilità che esso sia stato inviato è la stessa qualunque sia il crittogramma c in transito, poiché $P(M = m|C = c)$ dipende ora solo da m.

Il seguente teorema, dovuto a Shannon, costituisce una importante limitazione all'uso degli schemi di cifratura perfetti.

Teorema 3.1.1. *In un cifrario perfetto il numero delle chiavi deve essere maggiore o uguale al numero dei messaggi possibili.*

L'uso dei cifrari perfetti risulta quindi molto costoso e difficilmente realizzabile in pratica perché richiederebbe di dover conservare in sicurezza una chiave potenzialmente molto lunga che come abbiamo visto deve essere generata in modo casuale. Inoltre non è detto che la lunghezza del messaggio sia nota a priori.

Nonostante questi evidenti svantaggi, i cifrari perfetti sono molto interessanti e per chi richieda una sicurezza assoluta e come vedremo in seguito hanno anche una certa utilità pratica. Discuteremo ora la struttura, i pregi e i difetti del più importante di questi cifrari.

3.2 Lo Schema One-Time Pad (detto anche cifrario di Vernam's)

Nel 1917 Vernam brevettò un cifrario perfetto molto semplice e veloce che fu utilizzato a quanto si sa, per le comunicazioni diplomatiche tra Washington e Mosca durante la guerra fredda. Il nome del cifrario si riferisce familiarmente alla sequenza della chiave come se fosse scritta su un blocco di appunti (pad), e indica come tale sequenza venga progressivamente utilizzata per la cifratura e non sia riutilizzabile (one-time). Per utilizzare il cifrario dobbiamo anzitutto assumere che i messaggi, le chiavi e i crittogram-

mi siano sequenze binarie arbitrariamente lunghe, il che è comunque ovvio se l'informazione deve essere spedita sulla rete ed è memorizzata ed elaborata da calcolatori. Si noti che qualsiasi meccanismo di trasformazione del messaggio da codice originale (per esempio linguaggio naturale) a sequenza binaria (per esempio in codice ASCII) è pubblico, per cui tale trasformazione non ha nulla a che vedere con la segretezza della comunicazione.

Sia $a \oplus b$ l'OR esclusivo bit a bit di due stringhe binarie a e b, detto anche XOR. Lo schema di cifratura one-time pad è definito come segue:

1. sia l un intero positivo non nullo, lo spazio dei messaggi *Msg*, lo spazio delle chiavi *Key* e lo spazio dei testi cifrati *Critto*, sono l'insieme delle stringhe binarie di lunghezza l ossia $\{0,1\}^l$.

2. l'algoritmo di generazione della chiave Gen lavora scegliendo una stringa da Key secondo la distribuzione uniforme, cioè ogni stringa tra le 2^l stringhe dell'insieme Key è scelta con probabilità 2^{-l}

3. l'algoritmo di cifratura Enc lavora in modo tale che data una chiave $k \in \{0,1\}^l$ e un messaggio $m \in \{0,1\}^l$ il testo cifrato c sia $c := k \oplus m$

4. l'algoritmo di cifratura Enc lavora in modo tale che data una chiave $k \in \{0,1\}^l$ e un messaggio $m \in \{0,1\}^l$ il testo cifrato c sia $c := k \oplus m$

5. l'algoritmo di decifrazione Dec lavora in modo tale che data una chiave $k \in \{0,1\}^l$ e un messaggio cifrato $c \in \{0,1\}^l$ il testo in chiaro m sia $m := k \oplus c$

Osserviamo che vale $\text{Dec}_k(\text{Enc}_k(m)) = k \oplus k \oplus m = m$

Teorema 3.2.1. *Lo schema di cifratura one-pad è perfetto e impiega il numero minimo di chiavi.*

Dimostrazione. Occorre provare che:

$$P(M = m) = P(M = m | C = c)$$

Applicando la definizione di probabilità condizionale possiamo riscrivere il termine destro della formula come:

$$P(M = m|C = c) = \frac{P(M = m, C = c)}{P(C = c)} \tag{1}$$

Osserviamo che l'evento $\{M = m, C = c\}$ descrive la situazione in cui il mittente ha generato il messaggio m e lo ha cifrato come c. Per la definizione di XOR, fissato il messaggio, chiavi diverse danno origine a crittogrammi diversi, e ogni chiave può essere generata con probabilità $(\frac{1}{2})^n$. Dunque, fissato m risulta $P(C = c) = (frac12)^n$ per ogni c, quindi gli eventi M=m e C=c sono indipendenti e si ha

$$P(M = m, C = c) = (\frac{1}{2})^n \dot{P}(M = m) \tag{2}$$

Combinando le equazioni (1) e (2) si ha la tesi. La minimalità del numero di chiavi si ottiene dal Teorema 3.1.2 osservando che poiché le chiavi stesse sono sequenze arbitrarie di l bit si ha $|Key| = |Msg| = 2^l$ □

Capitolo 4

Cifratura a Chiave Privata e Pseudocasualità

In questo capitolo vengono definiti i criteri per definire la *bontà* di un generatore di numeri pseudocasuale. Viene presentata la suite del National Institute of Standards and Technology presentata nella Special Pubblication 800-22rev1a (Aprile 2010) intitolata *A Statistical Test Suite for Random and Pseudorandom Number Generators* che fornisce serie di test statistici per la prova e validazione di generatori di numeri pseudo casuali.

4.1 Introduzione

Nel precedente capitolo abbiamo presentato gli schemi crittografici di cui possiamo dimostrare la segretezza perfetta anche se l'avversario possedesse una potenza computazionale illimitata. In particolare in un cifrario perfetto i testi cifrati non contengono nessuna informazione sul testo in chiaro (supponendo la chiave segreta). Ci occupiamo ora di un altro concetto di sicurezza che è alla base dei moderni sistemi crittografici: la *sicurezza computazionale*. Gli schemi crittografici computazionalmente sicuri, non soddisfano la Definizione 3.1.1 e potrebbero essere forzati da un avversario con sufficiente

capacità computazionale e tempo illimitato. Tuttavia, sotto certe ipotesi, il tempo e il numero dei calcoli necessari per forzare tali schemi richiede più di una vita umana anche utilizzando il computer più potente e quindi permettono, di fatto, un livello di sicurezza sufficiente. Nonostante la sicurezza perfetta sia una condizione più forte della sicurezza computazionale, i cifrari perfetti come abbiamo visto nel precedente capitolo pongono limiti severi alla lunghezza delle chiavi e diverse problematiche relativamente allo scambio e alla segretezza delle stesse, pertanto occorre rinunciarvi se si vogliono ottenere schemi di crittografia utilizzabili in pratica.

4.2 Principi di Base di Sicurezza Computazionale

Kerckhoffs è conosciuto per aver affermato che gli algoritmi di crittografia debbano essere resi pubblici. Tuttavia egli formulò sei principi, tra i quali il seguente è rilevante per la nostra tesi:

> *Uno schema di cifratura deve essere praticamente, se non matematicamente, indecifrabile.*

In termini moderni, questo principio afferma che non è necessario utilizzare un cifrario perfetto, ma è sufficiente usare uno schema di cifratura che non può essere forzato in un ragionevole lasso di tempo e con una ragionevole probabilità di successo (nel linguaggio di Kerckhoffs uno schema che è praticamente indecifrabile). In termini pratici è sufficiente usare uno schema che può (in teoria) essere forzato, ma che non può essere forzato con una probabilità superiore a 10^{-30} in 200 anni usando il più veloce supercomputer disponibile. L'approccio computazionale presuppone di indebolire la nozione di sicurezza perfetta:

1. La sicurezza è da conservare solo contro avversari efficienti che lavorano in un ragionevole lasso di tempo;

2. Gli avversari possono aver successo con una probabilità veramente bassa (cioè sufficientemente bassa in modo tale che non riguardi ciò che realmente accade).

Per ottenere una teoria significativa, dobbiamo definire precisamente cosa significano le affermazioni precedenti. Ci sono due modi per farlo: l'approccio asintotico e quello concreto.

Approccio concreto: In questo approccio la sicurezza di uno schema crittografico è esplicitata limitando la massima probabilità di successo di un avversario che lavora per un tempo fissato. Siano t, ε due costanti positive con $\varepsilon \leq 1$. Uno schema è (t, ε)-sicuro se ogni avversario che lavora per un tempo al massimo t riesce a forzare lo schema con probabilità al più uguale a ε. Se misuriamo il tempo in cicli di CPU, si potrebbe scegliere t=280 cicli e $\varepsilon = 2^{-64}$. L'approccio concreto definisce schemi che siano (t, ε)-sicuri, ma mai semplicemente sicuri, perché la sicurezza dipende dalla capacità computazionale necessaria per una coppia di (t, ε) fissata.

Approccio asintotico: In questo approccio, sia il tempo dell'avversario che la sua probabilità di successo sono funzioni di certi parametri piuttosto che numeri concreti. Più precisamente, uno schema crittografico comprende un parametro di sicurezza che è un intero positivo n. Quando lo schema viene inizializzato, le parti scelgono il valore di n; questo valore è conosciuto anche da ogni avversario che attacchi lo schema. Sia il tempo che la probabilità di successo sono funzioni di n. Quindi:

1. Un algoritmo è efficiente se lavora in tempo polinomiale in n (abbrevieremo con PPT per indicare tempo probabilistico polinomiale). Ciò significa che per certe costanti a, h l'algoritmo funziona per un tempo $a \cdot n^h$. La sicurezza è richiesta solo verso avversari che lavorino per un tempo polinomiale.

2. Il concetto di piccola probabilità di successo significa con probabilità di successo tale che per ogni costante h, la probabilità di successo di

un avversario è più piccola di n^{-h} per valori sufficientemente grandi di n. Una funzione che cresce in questo modo è detta *trascurabile*.

Dunque uno schema è sicuro se *per ogni PPT un avversario riesce a forzare lo schema con probabilità trascurabile.* E' importante sottolineare che l'approccio computazionale garantisce la sicurezza solo per valori di n sufficientemente grandi. L'approccio asintotico ha il vantaggio di non dipendere da ipotesi sulla capacità computazionale dell'avversario. Il compito di determinare il valore del parametro di sicurezza n è complesso e dipende dallo schema in questione e da altre considerazioni. Nel seguito utilizzeremo l'approccio asintotico.

4.3 Cifratura Computazionalmente Sicura

Prima di presentare la definizione matematica di sicurezza computazionale per la cifratura a chiave privata, ridefiniamo la sintassi di uno schema di cifratura a chiave privata introducendo il parametro di sicurezza.

Definizione 4.1. Uno schema di cifratura a chiave privata è una terna di algoritmi probabilistici polinomiali (Gen, Enc, Dec) tali che:

1. L'algoritmo di generazione della chiave Gen prende in input il parametro di sicurezza 1^n e produce una chiave k. Supponiamo senza perdita di generalità che ogni chiave sia di lunghezza maggiore o uguale a n.

2. L'algoritmo di cifratura Enc prende in input una chiave k e un messaggio $m \in \{0,1\}^*$ e produce un testo cifrato c.

3. L'algoritmo di decifrazione Dec prende in input una chiave k, un testo cifrato c e produce un messaggio m. Assumiamo che Dec sia deterministico.

Si richiede che per ogni n e ogni chiave k e ogni $m \in \{0,1\}^*$ risulti:

$$\text{Dec}_k(\text{Enc}_k(m)) = m$$

Se (Gen,Enc,Dec) sono tali che per k generato da $\text{Gen}(1^n)$ l'algoritmo Enc_k è definito solo per messaggi $m \in \{0,1\}^{l(n)}$, diremo che (Gen, Enc, Dec) è uno schema di lunghezza fissata per messaggi di lunghezza $l(n)$.

Ci sono molti modi per definire la sicurezza di uno schema di cifratura a chiave privata che differiscono sostanzialmente per il tipo di capacità computazionale attribuito all'avversario. Ci occuperemo della sicurezza contro una forma debole di attacco, quello di tipo ciphertext-only, dove l'avversario che intercetta la comunicazione osserva un singolo messaggio senza avere nessun altro tipo di interazione né con il mittente né con il destinatario e la sua capacità computazionale è di tipo polinomiale. Non faremo invece nessuna supposizione riguardo le possibili strategie dell'avversario e ciò è molto importante per dare significatività alla nostra nozione di sicurezza, perché è impossibile prevedere tutte le possibili strategie. Date queste premesse, diremo intuitivamente che lo schema è sicuro se l'avversario non è in grado di dedurre nessuna informazione sul testo in chiaro dallo studio del testo cifrato.

La definizione di sicurezza semantica formalizza direttamente questa idea intuitiva ed è stata la prima definizione di sicurezza ad essere proposta. Sfortunatamente la definizione di sicurezza semantica è piuttosto complessa e di difficile comprensione. Esiste però una definizione equivalente, ma molto più semplice, che si basa sul concetto di *indistinguibilità*. Vediamo di formalizzare meglio questo concetto.

Indistinguibilità in presenza di una intercettazione

Sia $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ uno schema di cifratura a chiave privata, n un qualsiasi valore per il parametro di sicurezza e sia $\mathcal{A}$ un avversario. Definiamo l'esperimento di indistinguibilità in presenza di una intercettazione e lo indichiamo con $PrivKey_(A,\Pi)^{eav}$ come:

1. L'avversario $\mathcal{A}$ produce una coppia di messaggi m_0 e m_1 della stessa lunghezza.

2. L'algoritmo Gen produce una chiave con input 1^n e viene scelto a caso

un bit b nell'insieme $\{0,1\}$. Il testo cifrato c risultato della computazione $\text{ENC}_k(m_b)$ viene fornito all'avversario $\mathcal{A}$. Chiameremo *c testo cifrato di sfida.*

3. $\mathcal{A}$ produce un bit b'

4. L'output dell'esperimento è 1 se $b' = b$ e 0 altrimenti. Se $PrivKey^{eav}_{(A,\Pi)} = 1$ diremo che l'avversario $\mathcal{A}$ ha avuto successo.

Osserviamo che non c'è alcuna limitazione alla lunghezza dei messaggi m_0 e m_1 purché siano uguali (naturalmente poiché l'avversario ha capacità computazionale polinomiale, m_0 e m_1 dovranno avere lunghezza polinomiale). Se lo schema Π prevede una lunghezza fissa dei messaggi, sia essa $l(n)$, la definizione precedente deve essere modificata richiedendo che m_0 e $m_1 \in \{0,1\}^{l(n)}$.

Definizione 4.2. Uno schema $\Pi = (\text{GEN}, \text{ENC}, \text{DEC})$ ha cifratura indistinguibile in presenza di intercettazione se per tutti gli avversari $\mathcal{A}$ con capacità computazionale polinomiale esiste una funzione trascurabile *negl* tale che:

$$Pr[PrivKey_(\mathcal{A}, \Pi)^{eav} = 1] \leq \tfrac{1}{2} + negl(n)$$

In altre parole la definizione precedente afferma che *uno schema di cifratura è indistinguibile per intercettazione se qualsiasi avversario $\mathcal{A}$ potrebbe riuscire ad indovinare quale messaggio è stato cifrato con una probabilità maggiore di una quantità trascurabile di quella che avrebbe tentando a caso.*

Il seguente teorema fondamentale assicura l'equivalenza tra la nozione semantica di sicurezza e l'indistinguibilità.

Teorema 4.3.1. *Uno schema di cifratura a chiave privata è semanticamente sicuro se e solo se è indistinguibile da un avversario che intercetta il messaggio.*

Questo teorema ci permette di usare l'indistinguibilità come la nostra definizione di sicurezza.

4.4 Pseudocasualità

La nozione di pseudocasualità gioca un ruolo fondamentale nella crittografia in generale e nella crittografia a chiave privata in particolare. Banalmente una stringa pseudocasuale è una stringa che sembra essere generata in modo uniformemente casuale finché chi la osserva lavora in tempo polinomiale. Così come il concetto di indistinguibilità è un indebolimento di quello di sicurezza perfetta, così la pseudocasualità è un indebolimento della pura casualità. Tecnicamente nessuna stringa fissata può essere definita pseudocasuale (così come non ha molto senso riferirsi a nessuna stringa fissata come stringa casuale). Piuttosto, la pseudocasualità si riferisce ad una distribuzione di stringhe, e quando diciamo che una distribuzione D di stringhe di lunghezza l è pseudocasuale ciò significa che $\mathcal{D}$ è indistinguibile dalla distribuzione uniforme sulle stringhe di lunghezza l. La pseudocasualità può essere di aiuto nella costruzione di schemi di cifratura a chiave privata infatti, se un testo cifrato sembra casuale è chiaro che nessun avversario può avere da esso informazioni circa il testo in chiaro. In una certa misura, questa è l'esatta intuizione che sta alla base della sicurezza perfetta del one-time pad. In quel caso, il testo cifrato è distribuito uniformemente e non rivela nulla del testo in chiaro. Il one-time pad calcola lo XOR di una stringa casuale (la chiave) con il testo in chiaro. Se si usasse una stringa pseudocasuale come chiave, ciò potrebbe non essere rilevante per un osservatore con capacità computazionale polinomiale. Un *generatore di numeri pseudocasuali* è un algoritmo che parte da un piccolo valore iniziale detto *seme*, solitamente fornito come dato di ingresso, e genera una sequenza arbitrariamente lunga di numeri; questa a sua volta contiene una sottosequenza detta *periodo* che si ripete indefinitamente. In linea di principio un generatore è tanto migliore quanto più lungo è il periodo, perché è questo che dovrebbe essere successivamente utilizzato come sequenza casuale. Questi generatori possono però essere considerati *amplificatori di casualità* perché se innescati da un seme casuale di lunghezza m, fornito dall'utente o prodotto con uno dei metodi visti sopra nel capitolo precedente, generano una sequenza *apparentemente* casuale di lunghezza

$n \gg m$. Una inerente limitazione è che il numero di sequenze diverse che possono essere così generate è al massimo pari al numero di semi possibili, cioè 2^m nel caso binario, enormemente minore del numero complessivo 2^n delle sequenze lunghe n. Tali caratteristiche motivano per questi generatori l'appellativo di pseudo casuali. Torneremo nel prossimo capitolo sul problema di valutare un generatore di stringhe pseudocasuali. Possiamo formalizzare il concetto di stringa pseudocasuale richiedendo che un algoritmo polinomiale produca in output 1 con la stessa probabilità quando gli sia data in input una stringa puramente casuale o una stringa pseudocasuale. Un generatore pseudocasuale pertanto usa la piccola quantità di casualità per generare una lunga stringa pseudo casuale. Vediamo la definizione formale

Definizione 4.3. Sia n la lunghezza del seme e sia $l()$ una funzione polinomiale e sia $\mathcal{G}$ un algoritmo deterministico polinomiale tale che per ogni input $s \in \{0,1\}^n$ produca una stringa di lunghezza $l(n)$. Diremo che $\mathcal{G}$ è un generatore pseudocasuale se e solo se:

1. Per ogni n si ha $l(n) > n$

2. Per tutti i discriminanti $\mathcal{D}$, esiste una funzione trascurabile *negl* tale che

$$|Pr[\mathcal{D}(r) = 1] - Pr[\mathcal{D}(\mathcal{G}(s)) = 1| \leq negl(n)$$

   Dove r è scelto uniformemente a caso nell'insieme $\{0,1\}^{l(n)}$, il seme n è scelto uniformemente a caso nell'insieme $\{0,1\}^{l(n)}$. La funzione $l()$ è chiamata il fattore di espansione di $\mathcal{G}$.

Sottolineiamo che l'output di un generatore pseudocasuale è in verità molto lontano da essere casuale. Per esempio consideriamo il caso in cui $l(n) = 2n$ (cioè $\mathcal{G}$ duplica la lunghezza del suo input). La distribuzione uniforme su $\{0,1\}^{2n}$ è caratterizzata dal fatto che ognuna delle 2^{2n} stringhe possibili è scelta con probabilità pari a $2 - 2n$. D'altra parte consideriamo la distribuzione generata da $\mathcal{G}$. Poiché $\mathcal{G}$ riceve un input di lunghezza n, il

numero di stringhe differenti possibili è al più 2^n. Così la probabilità che una stringa casuale di lunghezza $2n$ sia nel range di $\mathcal{G}$ è al più

$$\frac{2^n}{2^{2n}} = 2^{-n}$$

cioè gran parte delle stringhe di lunghezza $2n$ non sono nell'output di $\mathcal{G}$. Ciò significa che è molto facile stabilire se una stringa è casuale o pseudo-casuale avendo a disposizione un tempo illimitato. Consideriamo infatti un algoritmo discriminatore $\mathcal{D}$ che lavora come segue: se in ingresso ha una stringa w produce 1 se se e solo se esiste $s \in \{0,1\}^n$ tale che $\mathcal{G}$(s)=w (questo calcolo può essere portato avanti calcolando $\mathcal{G}(s)$ per ogni $s \in \{0,1\}^n$ e questo può essere fatto perché le specifiche di $\mathcal{G}$ sono note, solo il seme è sconosciuto). Ora, se w era generata da $\mathcal{G}$, ciò porta a dire che $\mathcal{D}$ genera 1 con probabilità 1. Al contrario, se w è uniformemente distribuita in $\{0,1\}^{2n}$ allora la probabilità che esista un s con $\mathcal{G}(s) = w$ è al più 2^{-n} e così $\mathcal{D}$ genera 1 in questo caso con probabilità al più 2^{-n} , da cui:

$$|Pr[\mathcal{D}(r) = 1] - Pr[\mathcal{D}(\mathcal{G}(s)) = 1| \geq 1 - 2^{-n}$$

che è grande. Questo tipo di attacco è chiamato di *forza bruta* perché si limita a provare tutti i semi possibili. Il vantaggio di tale attacco è che è applicabile a tutti i tipi di generatori, indipendentemente da come lavorano.

Quanto detto mostra che la distribuzione generata da G non è quella uniforme. Tuttavia un algoritmo discriminatore che lavori in tempo polinomiale non può applicare la procedura precedente. Infatti se G è un generatore pseudo casuale, allora è garantito che non c'è nessuna procedura polinomiale che distingua tra stringhe casuali e pseudo casuali. Ciò significa che le stringhe pseudo casuali sono tanto valide quanto quelle casuali, purché il seme resti segreto e si considerino solo avversari che lavorino in tempo polinomiale. Dalla discussione precedente risulta inoltre che il seme deve essere lungo abbastanza affinché nessun algoritmo efficiente abbia il tempo di provare tutti i semi possibili. Tecnicamente, se n è sufficientemente grande, non è possibile che un algoritmo polinomiale possa cercare attraverso i 2^n semi possibili.

4.5 Costruzione di Schemi di Cifratura Sicuri

Siamo pronti ora a costruire uno schema di cifratura che abbia la proprietà di indistinguibilità. Questo schema sarà molto simile allo schema one-time pad tranne per il fatto che è utilizzata una stringa pseudocasuale piuttosto che una casuale. Poiché una stringa pseudocasuale sembra essere casuale se l'avversario lavora per un tempo polinomiale, si può dimostrare che lo schema è sicuro.

Costruzione 1. *Sia $\mathcal{G}$ un generatore pseudocasuale con fattore di espansione l (cioè $|\mathcal{G}(s)| = l(s)$) definiamo uno schema di cifratura a chiave privata per i messaggi di lunghezza l come segue:*

- GEN*: su imput 1^n sceglie $k \in \{0,1\}^n$ in modo uniformemente e produce questo come chiave.*
- ENC*: prende in input $k \in \{0,1\}^n$ e un messaggio $m \in \{0,1\}^{l(n)}$ e produce il testo cifrato $c := \mathcal{G}(k) \oplus m$*
- DEC*: prende in ingresso una chiave $k \in \{0,1\}^n$ e un testo cifrato $k \in \{0,1\}^{l(n)}$ e produce in output il testo in chiaro: $m := \mathcal{G}(k) \oplus c$*

Il seguente teorema garantisce la sicurezza di tale costruzione.

Teorema 4.5.1. *Se $\mathcal{G}$ è un generatore pseudocasuale, allora la costruzione 4.4.1 è uno schema di cifratura a chiave privata di lunghezza fissata che ha la proprietà di essere indistinguibile.*

Il punto cruciale della costruzione precedente è che la stringa di l bit $\mathcal{G}(k)$ può essere molto più lunga della chiave k. In particolare, usando lo schema costruito, è possibile cifrare un file che ha una lunghezza dell'ordine dei megabyte usando una stringa di soli 128 bit.

Capitolo 5

Test di Casualità

In questo capitolo vengono approfonditi i concetti matematici che sono alla base dei primi quattro test della suite del NIST.

5.1 Il NIST

Il National Institute of Standards and Technology (NIST) è un'agenzia del governo degli Stati Uniti d'America che si occupa dell'innovazione tecnologica. Fondato nel 1901, il NIST è uno dei più antichi laboratori di fisica della nazione e fa parte del Dipartimento del Commercio statunitense. Inizialmente si chiamava National Bureau of Standards (NBS) e la sua funzione è quella di promuovere l'innovazione e la competitività industriale degli Stati Uniti attraverso collaborazioni con l'industria per sviluppare standard, tecnologie e metodologie che favoriscano la produzione e il commercio e facendo avanzare la scienza degli strumenti di misura, gli standard, e la tecnologia in generale per migliorare la sicurezza economica e la nostra qualità della vita. Da oltre 100 anni, il National Institute of Standard and Technology ha di fatto contribuito a mantenere l'industria statunitense all'avanguardia e i suoi scienziati hanno dato grande contributo in diversi ambiti scientifici come per esempio nell'elaborazione di immagini scientifiche, nello studio dei nano

materiali oppure nella realizzazione di orologi atomici. Oggi innumerevoli prodotti e servizi si affidano in qualche modo sulla tecnologia, sulle misure e sugli standard forniti dal NIST. L'Information Technology Laboratory è il settore del NIST che si occupa in particolare delle tecnologie dell'informazione e ha al suo interno la Computer Security Division (CSD) il cui compito principale è fornire standard e strumenti per proteggere i sistemi di informazione contro le minacce alla riservatezza, all'integrità e alla disponibilità di informazioni e servizi. Al suo interno, il Cryptographic Technoloy Group (CTG) si occupa di costruire avanzati sistemi di autenticazione, infrastrutture a chiave pubblica, protocolli crittografici, conservazione sicura di chiavi e molto altro ancora. Man mano che sviluppa nuovi standard, consegne e linee guida, il CTG mira a sviluppare un toolkit crittografico globale che consenta alle agenzie governative statunitensi e a chiunque ne sia interessato di scegliere i componenti e le funzionalità di sicurezza crittografica per proteggere i loro dati, le loro comunicazioni e le loro operazioni. Il toolkit include attualmente una vasta gamma di algoritmi di crittografia, tecniche avanzate di sicurezza informatica, norme e linee guida, riguardanti i cifrari a blocchi, le firme digitali, il Message Authentication, la generazione di numeri casuali. Quest'ultimo argomento è particolarmente interessante per questa tesi. Nella NIST Special Pubblication 800-22*rev1a* (Aprile 2010) intitolata *A Statistical Test Suite for Random and Pseudorandom Number Generators* viene fornito un set di test statistici per la prova e validazione di generatori di numeri casuali (RNG) e pseudocasuali (PRNG). Questi test sono utili per determinare la deviazione di una stringa binaria da una che sia perfettamente casuale.

5.2 Generatori di Numeri Casuali e Pseudocasuali

Un generatore "perfetto" di numeri casuali è dato da una moneta "non truccata" con i lati etichettati da "0" e "1". Se lanciamo la moneta produr-

remo una sequenza di 0 e 1 perfettamente casuale in cui gli "0" e "1" sono uniformemente distribuiti. Ogni lancio avrà la probabilità di $\frac{1}{2}$ di essere uno "0" o un "1". Tutti gli elementi della sequenza saranno generati *indipendentemente* uno dall'altro e il valore dell'elemento successivo non può essere previsto, qualunque sia il numero degli elementi che è già stato prodotto. Sebbene l'uso di una moneta per scopi crittografici non sia praticabile, tuttavia l'output di tale generatore ideale è una sequenza perfettamente casuale che può essere utilizzata come benchmark per la valutazione di generatori di numeri casuali e pseudocasuali. Nel caso dei PRNG, se il seme è sconosciuto, il numero successivo nella sequenza deve essere imprevedibile nonostante la conoscenza dei precedenti numeri della sequenza. Questa proprietà è nota come "imprevedibilità in avanti". Inoltre non deve essere possibile prevedere il seme conoscendo un qualunque valore generato (proprietà questa nota come "imprevedibilità all'indietro"). Ciò significa che non deve essere evidente nessuna correlazione tra il seme e qualunque valore generato; ogni elemento della sequenza deve apparire come il risultato di un evento indipendente la cui probabilità è $\frac{1}{2}$. Per assicurare la imprevedibilità in avanti occorre fare molta attenzione a come viene ottenuto il seme. I valori prodotti da un PRNG sono completamente prevedibili se il seme e l'algoritmo di generazione della sequenza sono conosciuti. Poiché in molti casi l'algoritmo di generazione è pubblicamente noto, il seme deve essere tenuto segreto e non deve essere derivabile dalla sequenza che produce. Inoltre il seme stesso deve essere imprevedibile. Per generare il seme si potrebbe utilizzare un generatore di numeri casuali (RNG). Un RNG usa una sorgente non deterministica per produrre casualità. La sorgente consiste generalmente in una quantità fisica, come il rumore di un circuito elettrico, il succedersi degli istanti di tempo di qualche azione dell'utente (e.g., combinazioni di tasti premuti o movimenti del mouse), o effetti quantistici in un semiconduttore. Si usano anche combinazioni di tutti questi input. L'output di un RNG, può essere usato direttamente come un numero casuale o può essere dato in pasto ad un generatore di numeri pseudocasuale. Per essere usato direttamente (senza

ulteriori elaborazioni), l'output di un RNG deve soddisfare criteri molto stringenti di casualità misurati da test statistici per determinare se la sorgente fisica appare casuale. Per esempio il "rumore" elettronico può contenere una sovrapposizione di strutture regolari, come onde o altri fenomeni periodici. Inoltre alcune sorgenti fisiche (e.g., la data e l'ora) sono piuttosto prevedibili. Questi problemi possono essere ridotti combinando l'output di differenti tipi di sorgenti per ottenere l'input di un RNG. Tuttavia, l'output di un RNG può non essere ancora soddisfacente, se valutato da un test statistico. Inoltre la produzione di numeri casuali potrebbe essere troppo lenta e quindi non utilizzabile quando sia necessario costruire una grande quantità di numeri casuali. Per questo motivo è preferibile utilizzare generatori di numeri pseudocasuali. Un PRNG usa uno o più input e genera molteplici numeri "pseudocasuali". Gli output di un PRNG in generale sono funzioni deterministiche del seme, quindi la casualità perfetta è limitata alla generazione del seme e proprio per questo motivo si parla di generatori pseudocasuali. Stranamente sequenze pseudocasuali possono apparire molto più casuali di sequenze casuali prodotte da sorgenti fisiche. Se la sequenza pseudocasuale è costruita bene, ogni valore della sequenza è prodotto dal valore precedente con trasformazioni che appaiono introdurre ulteriore casualità. Una serie di queste trasformazioni può eliminare auto-correlazioni statistiche fra input e output. Così gli output di un PRNG possono avere migliori proprietà statistiche ed essere ottenuti addirittura più velocemente che in un RNG.

5.3 Testing dei Generatori Pseudocasuali di Numeri

Come abbiamo detto per verificare se una sequenza di numeri prodotta da un PRG soddisfa criteri di casualità, occorre sottoporla a test statistici. I test utilizzati nella suite del NIST si basano su una tecnica statistica chiamata test di ipotesi. Essa consiste nel verificare in termini probabilistici la validità di un'ipotesi riguardo a una certa caratteristica di una variabile aleatoria

definita sulla popolazione. Questa ipotesi è detta ipotesi nulla ed è indicata con H_0. L'ipotesi nulla è messa a confronto con una possibile alternativa che è indicata con H_a. Nel caso in questione, l'ipotesi nulla del test è che la sequenza testata è casuale, mentre l' ipotesi alternativa è che la sequenza test sia non casuale. Se i dati sono effettivamente casuali, allora respingere l'ipotesi nulla è un evento che si verifica in una piccola percentuale. Questa conclusione è detta errore di Tipo I. Se i dati sono non casuali, allora accettare l'ipotesi nulla è detto errore di Tipo II.

Per valutare il valore della caratteristica sotto esame (nel nostro caso la casualità della sequenza) si deve scegliere uno stimatore $\mathcal{S}$, detto anche *statistica test*, di cui conosciamo la distribuzione se supponiamo vera l'ipotesi nulla. A questo punto si decide un livello di errore α e si costruisce l'intervallo di confidenza $\mathcal{A}_0$ per lo stimatore $\mathcal{S}$, corrispondente al livello di errore scelto (si sceglie $\mathcal{A}_0$ in modo che $P(x \in \mathcal{A}_0) = 1 - \alpha$). Il valore α è quindi la probabilità di commettere un errore di Tipo I. Questa probabilità è spesso chiamata *livello di significatività del test*. Nel nostro caso,
alpha è la probabilità che il test indichi che la sequenza è non-random quando invece è realmente casuale. La probabilità di un errore di Tipo II è invece indicata da β. In accordo a quanto detto è importante sapere quanto vale 1-β, ovvero la probabilità di respingere (correttamente) una ipotesi falsa. Questo numero è detto *potenza del test*. Nel nostro caso, β è la probabilità che il test indichi che una sequenza è random quando invece non lo è. Un buon test dovrebbe avere contemporaneamente un piccolo α e un piccolo β, ovvero una grande potenza. Tuttavia non è possibile ridurre contemporaneamente queste due variabili che sono correlate, con β che cresce al diminuire di *alpha*. Ciò che si fa in pratica è fissare il valore di n e di α e quindi scegliere un valore critico t della statistica test che produca il più piccolo valore di β. Il valore di t sarà tale che $P(S > t|H_0) = \alpha$ o, analogamente, $P(S \leq t|H_0) = 1-\alpha$. Si noti che $P(S > t|H_0) = \alpha = P(rifiutare H_0|H_0 vera)$, per cui l'intervallo (t, ∞] è la zona di rifiuto di H_0 e di conseguenza $(-\infty$, t] è la zona di accettazione di H_0. In molti casi si effettuano poi i cosiddetti test a due code in cui il

valore critico del test è la quantità che soddisfa $P(-t \leq S \leq t|H_0) = 1 - \alpha$. In tal caso (-$\infty$, -t) $\cup$ (t , +∞) è la zona di rifiuto di H_0 mentre [-t, t] è la zona di accettazione di H_0. Un approccio alternativo è quello del calcolo di un P-*valore* (livello di probabilità) associato al valore calcolato della statistica test, s. Il P-*valore* associato alla statistica test calcolata s è la probabilità $P(S \geqslant s|H_0)$. Il P-*valore* indica quindi quanto probabile (valori alti) o improbabile (valori bassi) è l'eventualità di osservare esattamente il valore s della statistica test S sotto l'ipotesi nulla H_0. Infatti, se troviamo che P-*valore* ¡ α, cioè significa, usando la nostra notazione, che

$P(S \geqslant s|H_0) < \alpha = P(S > t|H_0)$ il che implica s ¿ t, e ciò significa che s è nella zona di rifiuto di H_0. Pertanto per rifiutare H_0 ci basta constatare che P-*valore* $\geqslant \alpha$

Al contrario, se troviamo che P-*valore* $\geqslant \alpha$, ciò significa, nella notazione dai noi usata, che P-*valore*=$P(S \geqslant s|H_0) \geqslant \alpha = P(S > t|H_0)$ ovvero s$\leqslant$t, il che comporta l'accettazione di H_0. Raggiungiamo tale conclusione senza dover necessariamente conoscere t, ci basta solo constatare che P-*valore* $\geqslant \alpha$.

La seguente tabella mette in relazione il reale stato (sconosciuto) dei dati a disposizione con la conclusione a cui si giunge usando la procedura di test.

Reale situazione	Conclusione	
	Accettare H_0	Accettare H_a (rifiuto di H_0)
I dati sono casuali (H_0 è vera)	Nessun errore 1-α	Errore di tipo I α
I dati non sono casuali (H_a è vera)	Errore di tipo II β	Nessun errore 1-β

Per quanto riguarda la scelta del livello di significatività, tipicamente α è scelto nel nell'intervallo [0.001 , 0.01].

- Un α di 0.001 indica che ci si aspetta che una sequenza su 1000 sia rifiutata dal test pur essendo casuale. Per un P-*valore*$\geqslant$0.001 una sequenza sarà considerata come random con una confidenza del 99.9%. Per un P-*valore*¡0.001 una sequenza sarà considerata non random con una confidenza del 99.9%.

- Un α di 0.01 indica che ci si aspetta che una sequenza su 100 sia rifiutata dal test pur essendo casuale. Per un P-*valore*$\geqslant$ 0.01 una sequenza sarà considerata come random con una confidenza del 99%. Per un P-*valore*¡0.01 una sequenza sarà considerata non-random con una confidenza del 99% (per esempio in questo documento α è stato scelto pari a 0.01).

Nella suite del NIST il livello di significatività è stato scelto pari all'1% cioè $\alpha = 0.01$ e prevede come statistica test la distribuzione normale e la distribuzione chi-quadrato (χ^2): Se la sequenza da testare è non random, la statistica calcolata cadrà nelle regioni estreme della distribuzione di riferimento. La distribuzione normale è usata per confrontare il valore della statistica test ottenuta dal PRGN con il valore atteso della statistica sotto l'ipotesi di casualità. La statistica test per la distribuzione normale è della forma $z = \frac{(x-\mu)}{\sigma}$, dove x è il valore statistico campione e μ e σ^2 sono la media e la varianza della statistica test. La distribuzione χ^2 è utilizzata per confrontare la bontà dell'adattamento delle frequenze osservate di una misura campione con le corrispondenti frequenze attese della ipotetica distribuzione. La statistica test è della forma $\chi^2 = \sum \frac{(o_i-e_i)^2}{e_i}$ dove o_i e e_i sono le frequenze osservate e attese delle occorrenze delle misure. Nel seguito faremo le seguenti ipotesi sulla sequenza binaria che deve essere testata:
Uniformità: In qualunque punto nella generazione della sequenza, la presenza di zero o uno è equiprobabile (con probabilità $\frac{1}{2}$). Il numero attesi di zero (o uno) in una sequenza di n elementi è quindi $\frac{n}{2}$.
Scalabilità: Ogni test applicabile ad una sequenza può essere applicato ad una sottosequenza estratta a caso. Se una sequenza è random, allora ogni sottosequenza estratta deve essere random. Quindi, ogni sottosequenza deve passare ogni test di casualità.
Consistenza: Il comportamento di un generatore deve essere consistente nei valori iniziali (seme).

5.4 La NIST Test Suite

La NIST Test Suite è un package statistico costituito da 15 test che sono stati sviluppati per provare la casualità di sequenze binarie (arbitrariamente lunghe) prodotte sia da generatori di numeri casuali o pseudocasuali di tipo hardware o software. Questi test si basano sui differenti tipi di non-casualità che possono esistere in una sequenza. Alcuni test sono scomponibili in una varietà di sotto-test. I 15 test sono:

- Frequency Test
- Frequency test a blocchi
- Runs Test
- Test per la più lunga sottosequenza di uno in un blocco
- Binary Matrix Rank Test
- Trasformata di Fourier discreta
- Non-overlapping Template Matching Test
- Overlapping Template Matching Test
- Maurer's Universal Statistical
- Test di complessità lineare
- Test seriale
- The Approximate Entropy Test
- Test somme cumulate
- Test di escursione casuale
- The Random Excursions Variant Test

L'ordine di applicazione di questi test è arbitraria. Tuttavia è consigliabile che il Frequency test sia applicato per primo, poiché questo fornisce la più basilare evidenza per la non-casualità, più precisamente la non-uniformità. Se questo test fallisce è molto probabile che anche gli altri test falliscano. (Nota: il più lungo test è il Linear Complexity test).

Frequency test
L'obiettivo di questo test è determinare se il numero di uno e di zero nella sequenza è approssimativamente lo stesso come ci si aspetterebbe da una sequenza veramente casuale. E' fortemente consigliato che la sequenza da provare sia di almeno 100 bit. Tutti i test successivi sono condizionati dall'aver superato questo test.

Test di Frequenza a blocchi
Lo scopo del test è di determinare se la frequenza di 1 in un blocco di M bit è approssimativamente pari a $\frac{M}{2}$ come dovrebbe essere se la sequenza fosse veramente casuale (si noti che per M=1 si ha il test precedente). E' consigliato che ogni sequenza da testare sia di almeno 100 bit. Indicato con N il numero dei blocchi che non si sovrappongono ($N = \lfloor n/M \rfloor$) si ha $n \geqslant MN$. L'ampiezza di M dovrebbe essere scelta in modo che $M \geqslant 20$, $M > 0.01n$ e $N < 100$.

Runs Test Una run di lunghezza k consiste di esattamente k bit identici ed è limitata prima e dopo da un bit del valore opposto. Lo scopo del test è di determinare se il numero di run di 1 e di 0 delle varie lunghezze è quello atteso per una sequenza casuale. In particolare, questo test determina se l'oscillazione tra questi 0 e 1 è troppo veloce o troppo lenta. Anche in questo caso il test è attendibile se n$\geqslant$100.

Longest Run of Ones in a Block
L'obiettivo di questo test è determinare se la lunghezza della più lunga run

di 1 della sequenza da testare è consistente con la lunghezza della più lunga run di 1 che ci si aspetterebbe in una sequenza casuale. Si noti che una irregolarità nella lunghezza prevista della più lunga run di uno implica che c'è anche una irregolarità nella più lunga run di zero. Perciò è necessario solo il test per gli uno. E' raccomandato che ogni sequenza testata consista di un numero di bit come specificato nella seguente tabella:

Minimo n	M
128	8
6272	128
750.000	104

Binary Matrix Rank Test
L'obiettivo del test è di controllare la dipendenza lineare tra sottostringhe della sequenza originale di una determinata lunghezza.

Test Trasformata di Fourier discreta
Questo test misura i picchi di altezza nella Trasformata di Fourier discreta della sequenza. Il proposito di questo test è controllare caratteristiche periodiche (i.e., pattern ripetitivi che sono vicini uno all'altro) nella sequenza testata il che indicherebbe una deviazione dall'ipotesi di casualità.

Non-overlapping Template Matching Test
Questo test individua il numero di occorrenze di un di una stringa obiettivo specificata. Il proposito è di individuare generatori che producono troppe occorrenze di un pattern non periodico dato. Per questo test si usa una finestra di m bit per trovare uno specifico pattern di m bit. Se il pattern non è trovato, la finestra scorre di un bit. Se il pattern è trovato la finestra è resettata al bit dopo il pattern trovato e la ricerca riprende.

Overlapping Template Matching Test
La differenza tra questo test e il test precedente è che se il pattern è trovato

la finestra scorre di un solo bit prima che la ricerca riprenda.

Maurer's "Universal Statistical" Test
L'obiettivo del test è controllare se la sequenza può essere significativamente compressa senza perdita di informazione. Una sequenza che possa essere compressa è considerata essere non-random.

Linear Complexity Test
L'obiettivo del test è determinare se la sequenza è abbastanza complessa da essere considerata random.

Serial Test
Il focus del test è la frequenza di tutti i possibili pattern di m bit che si sovrappongono attraverso l'intera sequenza. L'obiettivo di questo test è determinare se il numero di occorrenze dei 2^m pattern di m bit che si sovrappongono è lo stesso che si avrebbe in una sequenza casuale. Le sequenze casuali presentano una certa uniformità, cioè ogni pattern di m bit ha la stessa probabilità di apparire di ogni altro. Si noti che per m=1 il Serial Test equivale al Monobit test con i due pattern 0 e 1.

Approximate Entropy Test Come nel Serial Test il focus del test è la frequenza di tutti i possibili pattern di m bit che si sovrappongono attraverso l'intera sequenza. L'obiettivo del test è confrontare la frequenza di blocchi di lunghezza m e m+1 che si sovrappongono.
Cumulative Sums (Cusum) Test
L'obiettivo di questo test è determinare se la somma cumulata delle sequenze parziali nella sequenza iniziale è simile a quella di una sequenza realmente casuale.

Random Excursions Test
L'obiettivo del test è di verificare se il numero di visite di un determinato

stato all'interno di un ciclo è molto diverso da quello che ci aspetteremmo in una sequenza casuale.

Random Excursions Variant Test
L'obiettivo del test è contare il numero di volte che un particolare stato è visitato in una somma cumulata.

Descriveremo ora in dettaglio i primi quattro test che sono stati da me implementati come plugin del portale Cryptool.

5.5 Implementazione dei Test

Frequency Test
Questo test si basa sul

Teorema 5.5.1 (teorema centrale del limite)**.** *Sia $x_1, x_2, \ldots, x_n$ una successione di variabili aleatorie indipendenti e identicamente distribuite con la stessa media μ e la stessa varianza σ^2. Allora la distribuzione della variabile aleatoria*

$$y = \frac{\overline{x}-\mu}{\sigma}\sqrt{n}$$

dove

$$\overline{x} = \frac{\sum_{j=1}^{n} x_j}{n}$$

è la media aritmetica degli x_j, tende alla distribuzione normale standard per $n \to +\infty$. Ovvero per ogni $x \in \Re$ si ha:

$$\lim_{n\to+\infty} P(y \leqslant z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{t^2}{2}} dt$$

Ora

$$y = \frac{\overline{x}-\mu}{\sigma}\sqrt{n} = \frac{\frac{\sum_{j=1}^{n} x_j - \mu}{n}}{\sigma}\sqrt{n} = \frac{\sum_{j=1}^{n} x_j - n\mu}{n\sigma}\sqrt{n} = \frac{\sum_{j=1}^{n} x_j - n\mu}{\sqrt{n}\sigma}$$

Posto σ=1 e $\mu = \frac{1}{2}$, l'espressione precedente diventa

$$y = \frac{2\sum_{j=1}^{n} x_j - n}{\sqrt{n}} \tag{1}$$

Siano ora $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ la sequenza di variabili casuali che formano la chiave. Indicato con

$$X_i = 2\epsilon_i - 1$$

e con

$$S_n = X_1 + X_2 + \ldots + X_n$$

si ha

$$S_n = X_1 + X_2 + ... + X_n = 2(\epsilon_1 + \epsilon_2 + ... + \epsilon_n) - n \tag{2}$$

Se sostituiamo nella formula (1) gli $x_i = \epsilon_i$, si ottiene:

$$y = \frac{S_n}{\sqrt{n}}$$

Indicato con

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{t^2}{2}} dt$$

Dal Teorema centrale del limite si ottiene per $z \geqslant 0$

$$P(\frac{S_n}{\sqrt{n}} \leqslant z) = \Phi(z)$$

e, passando al valore assoluto di S_n

$$P(\frac{|S_n|}{\sqrt{n}} \leqslant z) = 2\Phi(z) - 1$$

Se assumiamo come statistica test

$$S = \frac{|S_n|}{\sqrt{n}}$$

e quindi il suo valore calcolato è

$$s = \frac{|X_1 + X_2 + ... + X_n|}{\sqrt{n}}$$

allora
$P-valore = P(S \geqslant s) = 1-(2\Phi(|s|)-1) = 2[1-\Phi(|s|)]$ Con il cambiamento di variabile $u = \frac{t}{\sqrt{2}}$ da cui $dt = \sqrt{2}du$ si ha

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{t^2}{2}} dt = \frac{\sqrt{2}}{\sqrt{2\pi}} \int_{-\infty}^{\frac{z}{\sqrt{2}}} e^{-u^2} du = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\frac{z}{\sqrt{2}}} e^{-u^2} du$$

da cui

$$\Phi(-z) = \frac{1}{\sqrt{\pi}} \int_{\frac{z}{\sqrt{2}}}^{\infty} e^{-u^2} du$$

Per simmetria si ha poi che $\Phi(-z) = 1 - \Phi(z)$ e quindi

$$P - valore = P(S \geqslant s) = 2[1 - \Phi(|s|)] = erfc\left(\frac{|s|}{\sqrt{2}}\right)$$

ove

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_{z}^{\infty} e^{-u^2} du$$

Nell'implementazione del Frequency test si calcola innanzi tutto

$$S_n = X_1 + X_2 + \ldots + X_n \text{con} X_i = 2\epsilon_i - 1 \tag{5.1}$$

e quindi

$$s = \frac{|S_n|}{\sqrt{n}}$$

Infine il P-*value* è dato da

$$erfc\left(\frac{s}{\sqrt{2}}\right)$$

E' raccomandato che la sequenza test sia formata da un minimo di 100 bit. Se il P-*value* è piccolo (< 0.01) ciò è dovuto a $|S_n|$ o $|s|$ troppo grandi. Un grande valore positivo di $|S_n|$ indica la presenza di troppi 1, mentre un grande valore negativo di $|S_n|$indica un numero troppo elevato di 0.

Frequency Test a Blocchi

Il test cerca di rilevare localizzate deviazioni dalla frequenza ideale del 50% di 1 scomponendo la sequenza da testare in un certo numero di sottosequenze che non si sovrappongono l'una con l'altra e applicando il test chi-quadrato. Piccoli P-*value* indicano una grande differenza con l'uguale proporzione di 1 e 0 in almeno una delle sottostringhe. Per ogni sottostringa si calcola la proporzione di uno e una statistica chi-quadrato confronta le proporzioni delle sottostringhe con il risultato ideale di $\frac{1}{2}$. La statistica di riferimento è una distribuzione chi quadrato con n gradi di libertà, dove n è la lunghezza della stringa. Suddividiamo quindi la sequenza da testare di lunghezza n in $N = \lfloor \frac{n}{M} \rfloor$ sottosequenze disgiunte di lunghezza M. La quantità

$$\pi_i = \frac{\sum_{j=1}^{M} \epsilon_{(i-1)M+j}}{M} i = 1, ..., N \tag{5.2}$$

rappresenta la media dei valori di ogni sottosequenza.
La somma

$$X^2 = 4M \sum_{i=1}^{N} \left[\pi_i - \frac{1}{2} \right]^2$$

ha una distribuzione di chi-quadrato con N gradi di libertà. Il corrispondente P-*value* è:

$$P{-}valore = P(X^2 \geqslant \chi^2) = \frac{\int_{\chi^2}^{\infty} e^{-\frac{u}{2}} u^{\frac{N}{2}-1} du}{\Gamma\left(\frac{N}{2}\right) 2^{\frac{N}{2}}} = \frac{\int_{\frac{\chi^2}{2}}^{\infty} e^{-u} u^{\frac{N}{2}-1} du}{\Gamma\left(\frac{N}{2}\right)} = igamic\left(\frac{N}{2}, \frac{\chi^2}{2}\right)$$

dove

$$\Gamma(z) = \int_0^{\infty} e^{-t} t^{z-1} dt$$

e

$$igamic(a,x) = \frac{\Gamma(a,x)}{\Gamma(a)}$$

Runs Test

Questo test si focalizza sulle "run" definite come sottostringhe di 1 o 0 consecutivi e considera se l'oscillazione fra tali sottostringhe omogenee è troppo veloce o troppo lenta. La statistica di riferimento è V_n che è la distribuzione del numero totale di runs, cioè

$$V_n = \sum_{k=1}^{n-1} r(k) + 1 \text{ dove } r(k) = 0 \text{ se } \epsilon_k = \epsilon_k + 1 \text{ e } r(k) = 1 \text{ se } \epsilon_k \neq \epsilon_k + 1$$

e si pone

$$\pi = \sum_j \frac{\epsilon_j}{n}$$

(dal Frequency test si ricava che $|\pi - \frac{1}{2}| \leqslant \frac{2}{sqrtn}$). Per il teorema del limite centrale si ha che

$$\lim_{n\to\infty} P\left(\frac{|V_n - 2n\pi(1-\pi)|}{2\sqrt{n}\pi(1-pi)} \leqslant z)\right) = \Phi(z)$$

Si ricava il P-value come nel frequency test e si ottiene

$$P_{valore} = erf\left(\frac{|V_n - 2n\pi(1-\pi)|}{2\sqrt{n}\pi(1-pi)}\right) \tag{5.3}$$

Test for the Longest Run of Ones in a Block

La stringa da testare è suddivisa in N sottosequenze di M bit ciascuna. Nella h-esima sottostringa si scelgono K+1 classi (in dipendenza da M). Per ognuna di queste sottosequenze si valutino le frequenze $\upsilon_0, \upsilon_1, ..., \upsilon_k$ ($\upsilon_0 + ... + \upsilon_k = N$ cioè il valore calcolato della più lunga run di uno in ognuna di queste sottostringhe appartenente a ciascuna delle K+1 classi scelte). Si può dimostrare che se ci sono r uno e M-r zero nel blocco m-esimo, la probabilità condizionale che la più lunga stringa di uno υ in questo blocco sia minore o uguale ad m ha la seguente forma:

$$P(\upsilon \leqslant m|r) = \frac{1}{\binom{M}{r}} \sum_{j=0}^{U} (-1)^j \binom{M-r+1}{j} \binom{M-j(m+1)}{M-r}$$

dove

$$U = min\left(M - r + 1, \left[\frac{r}{m+1}\right]\right)$$

cosicchè

$$P(\upsilon \leqslant m) = \frac{1}{\binom{M}{r}} \sum_{r=0}^{M} \binom{M}{r} P(\upsilon \leqslant m|r) \frac{1}{2^M} \qquad (*)$$

Prendiamo come statistica test la seguente

$$X^2 = \sum_{i=0}^{K} \frac{(\upsilon_i - N\pi_i)^2}{N\pi_i} \qquad (5.4)$$

dove le probabilità teoriche sono date dalla formula (*). La statistica test, sotto le ipotesi di casualità, ha la distribuzione approssimata χ^2 con K gradi di libertà. Analogamente a quanto fatto nel test di frequenza a blocchi il P-*value* è dato da:

$$P - valore = \frac{\int_{\chi^2}^{\infty} e^{-\frac{u}{2}} u^{\frac{K}{2}-1} du}{\Gamma\left(\frac{K}{2}\right) 2^{\frac{K}{2}}} = igamic\left(\frac{N}{2}, \frac{X^2}{2}\right) \qquad (5.5)$$

Capitolo 6

Crittografia: un Approccio Didattico

In questo capitolo viene illustrato il progetto Cryptool e i risultati ottenuti dall'implementazione e inserimento in questo progetto dei primi quattro test della suite del NIST. Viene infine presentata una breve attività di laboratorio per la scuola media basata sul cifrario di Cesare e sull'utilizzo del portale.

6.1 Crittografia e Didattica

Nei capitoli precedenti abbiamo più volte sottolineato quanto essa sia quotidianamente adoperata, magari inconsapevolmente, da chiunque utilizza servizi come il bancomat, la posta elettronica o il commercio on-line, o anche semplicemente per assistere a programmi televisivi a pagamento. Solo per citare un altro esempio, pensiamo che dal 1997 la cosiddetta legge Bassanini sancisce che un documento informatico dotato di firma digitale ha lo stesso valore legale di un tradizionale documento cartaceo con sottoscrizione autografa. La firma digitale, o firma elettronica qualificata (che non è la firma autografa digitalizzata, cioè la rappresentazione digitale di un'immagine cor-

rispondente alla firma autografa), è basata sulla tecnologia della crittografia a chiavi asimmetriche ed è un sistema di autenticazione di documenti digitali, in genere contenuta dentro una smart card (simile ad un bancomat), cioè una card che contiene un microprocessore. La firma digitale attesta la volontà del titolare della chiave privata di sottoscrivere il documento informatico e quindi di assumere la responsabilità del suo contenuto. Durante la mia esperienza di insegnamento (maturata da oltre dieci anni, dalle scuole medie ai Licei), ho potuto constatare con dispiacere quanto spesso la Matematica sia considerata da tanti studenti una materia noiosa, difficile, distante dai problemi quotidiani e priva di implicazioni pratiche. Gli alunni delle nostre scuole dell'obbligo e non solo di quelle, sono tutti nativi digitali, nelle loro case e nelle loro camerette, infatti, i media digitali sono sempre più presenti insieme alle esperienze di intrattenimento, socializzazione e formazione che vengono mediate e vissute attraverso Internet e i social network, oltre che per mezzo delle consolle per videogiochi. Alcuni studi condotti in Asia hanno mostrato che i ragazzi di oggi hanno un cervello che è più percettivo e meno simbolico rispetto a quello, per capirci, dei loro genitori. Si tratta di bambini e ragazzini davvero multitasking, in grado di distribuire l'attenzione su 4-5 dispositivi allo stesso tempo: studiano, ascoltano la musica, rispondono agli sms e guardano Facebook sul pc, contemporaneamente. Di fronte a questa netta superiorità della generazione digitale, cambiano anche i modi dell'apprendimento: spesso i ragazzi imparano di più attraverso il gioco che non facendo batterie di esercizi come invece eravamo abituati noi alla loro stessa età. Diventa quindi sempre più importante cercare nuove vie per catalizzare l'attenzione degli studenti, per facilitare l'apprendimento di concetti che sono diventati per loro molto più difficili e faticosi da imparare di quanto lo erano per noi. Alcune esperienze in questo senso sono le competizioni matematiche che ogni anno coinvolgono migliaia di studenti, come i Campionati Internazionali di Giochi Matematici organizzati dalla Università Bocconi di Milano, le Olimpiadi di Matematica patrocinati dall'Unione Matematica Italiana o le Olimpiadi di Informatica promosse dal Ministero dell'Istruzione, dell'U-

niversità e della Ricerca. Numerosi sono anche i progetti e le iniziative di utilizzo della crittografia a scopo didattico, rivolte soprattutto agli studenti delle scuole superiori e dell'Università. Per esempio, le CryptoWars che sono un'iniziativa a scopo divulgativo del Laboratorio di Matematica Industriale e Crittografia (Univiversità di Trento). Consistono in una gara nazionale che coinvolge singoli appassionati e team che si sfidano tra loro creando codici e rompendo quelli degli altri. In Gran Bretagna è giunto all'undicesima edizione il National Cypher Challenge, la competizione organizzata dalla University of Southampton School of Mathematics che ha visto impegnati oltre 6200 ragazzi provenienti da 725 scuole del territorio britannico. Si tratta di una gara nazionale di crittografia dove i giovani studenti vengono chiamati a decifrare alcuni messaggi postati online in un livello di difficoltà sempre crescente. Se gli stili di apprendimento cambiano non è possibile non tenerne conto nell'attività di insegnamento. Una scuola che dicesse semplicemente: "Studia che la matematica perché ti servirà", scordandosi che i ragazzi vivono nel presente, una scuola che demonizza videogiochi, MP3, social network e quant'altro fa parte della vita quotidiana dei nostri ragazzi, invece di capirli ed utilizzarli per quello che sono (e cioè esempi di uso vivo della Matematica), è destinata ad essere perdente e ad allontanare sempre di più i ragazzi dal desiderio di imparare. Un approccio possibile è quello di inserire la matematica scolastica in un contesto più ampio. In un articolo intitolato "Perchè la Matematica scolastica è noiosa e così poco interessante" Mario Valle (Visualization expert al Swiss National Supercomputing Centre, che da alcuni anni sviluppa progetti per diffondere la conoscenza delle connessioni tra le tecniche di visualizzazione, il calcolo e l'insegnamento) scrive:

> Occorre preparare la mente e stimolare l'interesse, motivando gli studenti attraverso casi concreti che mostrino cosa "serve" tutta la fatica della Matematica. Non è questione di proporre esempi concreti solo per far comprendere concetti astratti, che tra l'altro verrebbero limitati dalla concretezza dell'esempio, è far vedere e apprezzare la bellezza del panorama della Matematica

avvicinandola da angoli inusuali.

E' proprio nella ricerca di questi angoli inusuali da proporre ai miei studenti che in questa tesi e nel mio progetto di tirocinio mi sono voluta occupare di implementare alcuni test di casualità nell'ambito di un portale di crittografia che potrebbe essere appunto utilizzato per realizzare laboratori didattici nelle scuole.

6.2 Il Portale CrypTool

Il portale CrypTool riunisce differenti iniziative del progetto CrypTool che sviluppa i programmi di e-learning più diffusi al mondo nell'ambito della crittografia e della crittoanalisi. Consiste di cinque differenti sottoprogetti (CT1, CT2, JCT, CTO) che sono correlati al software Cryptool in vari aspetti e per differenti obiettivi.

- CrypTool1 (CT1) è stata la prima versione di CrypTool. E' stato rilasciato nel 1998 e permette di sperimentare differenti algoritmi crittografici. CT1 ha due successori.
- CrypTool2 (CT2) fornisce visual programming e esecuzione di procedure crittografiche in cascata.
- JCrypTool (JCT) che è platform-independent.
- CrypTool-Online (CTO) è stato rilasciato nella primavera del 2009. Questo tool permette di provare differenti algoritmi utilizzando browser e/o smartphone.

(Un altro sottoprogetto è il Mistery Twister C3 che fornisce una serie di indovinelli con i quali si possono testare le proprie conoscenze in ambito crittografico).

Il portale CrypTtool può essere utilizzato per aumentare le conoscenze di crittografia di chiunque sia interessato. Tutti i programmi sono open source

e tutti coloro che vogliono sviluppare parti del progetto sono ben accolti. E' un ambiente ideale per imparare sia la storia della crittografia che le ultime frontiere e le loro applicazioni. Può essere utilizzato per lo studio individuale o per l'approfondimento. Molti enti e organizzazioni hanno contribuito allo sviluppo del progetto CrypTool, i principali citati nel sito sono:

- Università di Siegen (Germania): ha contribuito ad estendere il numero di features del progetto che è stato presentato in un evento pubblico in cui ha ricevuto il premio come "Selected Landmark" nell'ambito del progetto "Germannia terra di idee" che premia la creatività e le iniziative tedesche in tutto il mondo;

- Università di Duisburg-Essen (Germania): è entrata a far parte del progetto dal 2007 con diverse committenze. In particolare hanno partecipato all'hosting del sito ufficiale e delle pagine degli sviluppatori del CrypTool2 e di tutto il materiale ad esso correlato. Fanno parte di questa Università il core architecture team del progetto CrypTool2;

- L'Università della Ruhr a Bochum: si è occupata della guida in linea e della valutazione dell'interfaccia utente. Inoltre hanno fatto da supervisione agli studenti che hanno integrato funzionalità in CrypTool2 e ora amministrano il sito del MTC3;

- Università Kassel: il Prof. Arno Wacker, direttore del dipartimento di Ingegneria del Software, è il Technical leader del progetto CrypTool2;

- Deutsche Bank: inizialmente il progetto è stato sviluppato nell'ambito dei corsi di formazione per la sensibilizzazione alla sicurezza informatica. Successivamente Deutsche Bank decise di consegnare CrypTool alla comunità open source;

- L'Università Tecnica di Darmstadt (Germania): si è occupata dell'hosting del primo sito web di Cryptool. Inoltre hanno fornito il primo repository del codice sorgente del Cryptool1.x. Oggi sono più coinvolti nel progetto JCrypTool;

- Centro di Ricerca per la Sicurezza Avanzata di Darmstadt: il gruppo di ricerca in Crittografia e Computer Algebra ha sviluppato una libreria in Java che è utilizzata in JCrypTool;

- Politecnico di Varsavia (Polonia): ha tradotto in polacco sia il software che il sito web del progetto;

- Consiglio Superiore della Ricerca Scientifica (Spagna): ha tradotto il software e il vecchio sito web in spagnolo;

- Università di Klagenfurt (Austria): il gruppo di ricerca per la sicurezza contribuisce alla certificazione di qualità come parte del beta testing team. Forniscono inoltre idee e riscontri rispetto a ulteriori sviluppi in ambito accademico;

- Università di Hagenberg (Austria): ha contribuito al progetto JCrypTool fin dal 2007 implementando procedure e protocolli.

Le funzioni sviluppate all'interno del portale sono moltissime, qui indico molto brevemente soltanto quelli che ho citato precedentemente o che sono particolarmente noti. Per quanto riguarda i cifrari classici sono implementati il cifrario di Atbash, il cifrario di Cesare e Vigenere, solo per citarne alcuni. Tra i cifrari moderni sono implementati il 3DES, l'AES, il DES, l'RSA e il RSA-DES. Per quanto riguarda le tecniche di crittanalisi sono sviluppati gli attacchi di forza bruta agli algoritmi 3DES, AES, RSA, al cifrario di Cesare e moltissimi altri. Sono implementati altresì alcuni generatori di numeri pseudocasuali e alcuni test (Frequency Test, Runs Test), ma soltanto nell'ambito del progetto CrypTool1.

6.3 Il Sottoprogetto CrypTool2

Il progetto CrypTool2 è un programma open source che fornisce una interfaccia grafica innovativa per sperimentare tecniche crittografiche. Permette

anche di visualizzare e controllare il flusso di lavoro degli algoritmi per consentire una manipolazione e interazione intuitiva con le funzioni crittografiche. Per questo motivo l'ho scelto nell'ambito della mia tesi e tirocinio. CrypTool2 fornisce una grande varietà di strumenti per analizzare cifrari moderni e classici. La guida in linea comprende sia istruzioni circa l'utilizzo delle funzioni sia richiami teorici e riferimenti per successivi approfondimenti. Non sono implementati invece nessun generatore di numeri casuali né nessun test per la loro verifica. Il programma gira sotto Windows sia a 32 che a 64 bit e necessita di Microsoft .NET Framework 4.0. Per accedere al codice sorgente è necessario utilizzare SVN (in particolare io ho installato TortoiseSVN come suggerito dalla pagina wiki del progetto). Per scaricare nel mio computer l'intera repository non è stato necessario registrarsi. Il codice sorgente è scritto in C#, che è un linguaggio di programmazione orientato agli oggetti sviluppato da Microsoft all'interno dell'iniziativa .NET. Compilare è piuttosto semplice: basta andare nella `trunk`
directory e aprire CrypTool 2.0.sln. A questo punto si apre l'ambiente di sviluppo integrato Visual Studio e, dopo aver scelto i setting opportuni, si può compilare la soluzione e successivamente avviarla attraverso il menu Debug.

Il progetto ha anche una pagina wiki che fornisce diverse informazioni, per esempio le linee guida per gli sviluppatori e alcuni Howtos, informazioni per partecipare ai gruppi di discussione via Skype o mailing list e anche una pagina in cui si trovano proposte per compiti da far svolgere agli studenti. Fra questi progetti, mi ha particolarmente interessato quello relativo allo sviluppo di test per la verifica della casualità di un generatore di numeri pseudo casuali perché era un argomento di cui non avevo mai sentito parlare fino ad ora e mi interessava conoscere.

6.4 Implementazione dei Plugin in CrypTool

Per l'implementazione dei plugin è fornito un template che contiene diverse parti commentate e contrassegnate dalla parola chiave Howto. Questi

sono suggerimenti per indicare cosa si deve cambiare per adattare lo scheletro del plugin alla propria implementazione. Ogni plugin contiene un file in cui è si trova il codice sorgente vero e proprio, un file in cui vi è il codice per la definizione dei setting (che sono i parametri che possono essere settati dall'utente per il funzionamento del plugin) e un file xml di documentazione che viene generato automaticamente dai commenti inseriti nel codice. I plugin sono raggruppati nelle seguenti ComponentCategory: Classic Cypher, Modern Ciphers, Steganography, Hash Function, Cryptanalysis, Protocols e Tools. All'interno della categoria Tools vi sono le seguenti sottocategorie: Boolean, Data Flow, Data input/output e Misc.

Ho realizzato essenzialmente 4 plugin che implementano gli omonimi algoritmi di verifica per un generatore di numeri pseudo casuali che sono stati descritti nel capitolo 5. Nel dettaglio essi sono Frequency-Test, blockFrequency, Runs-Test e LongestRun-Test. Ho scelto di includere questi plugin nella categoria Tools Misc. All'avvio il programma si presenta con la seguente schermata: Abbiamo detto in precedenza che CrypTool2 permette di

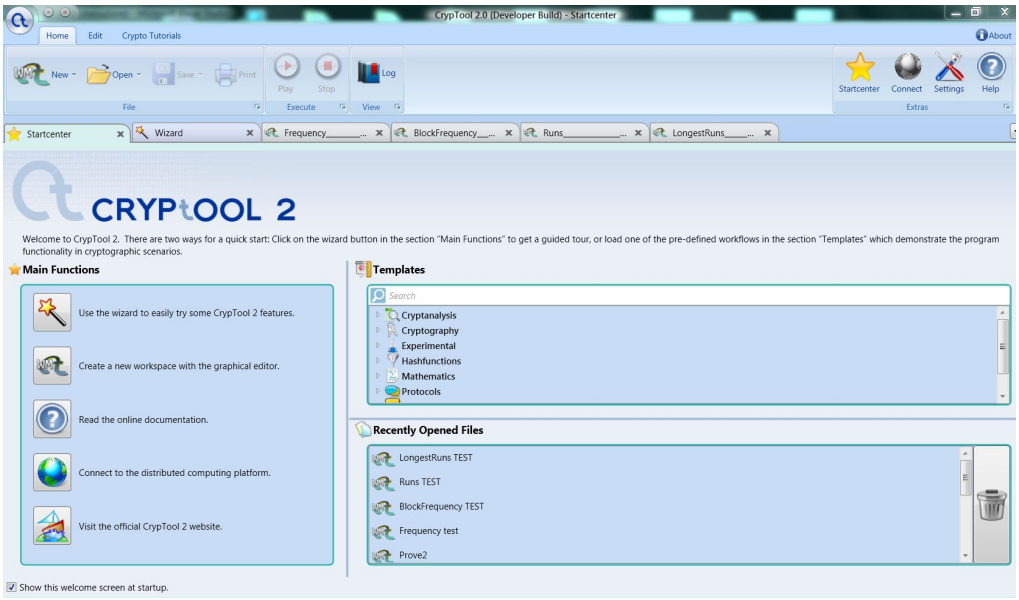

Figura 6.1 CrypTool2

visualizzare gli algoritmi e il loro flusso di lavoro. I plugin sono infatti rappre-

sentati attraverso rettangoli che hanno diverse modalità di visualizzazione: Presentation (che è la visualizzazione di default), Settings (che contiene i parametri che sono definiti nel file sorgente Settings) Log (in cui vengono mostrati errori o notifiche), Data (dove sono indicati i dati in ingresso e in uscita del plugin) e Help (che rimanda al file di documentazione xml). Si può passare da una rappresentazione all'altra cliccando sull'icona in alto a sinistra. Per esempio il Frequency-Test è così rappresentato nella visualizzazione Data. Questi rettangoli possono essere ridimensionati cliccando e

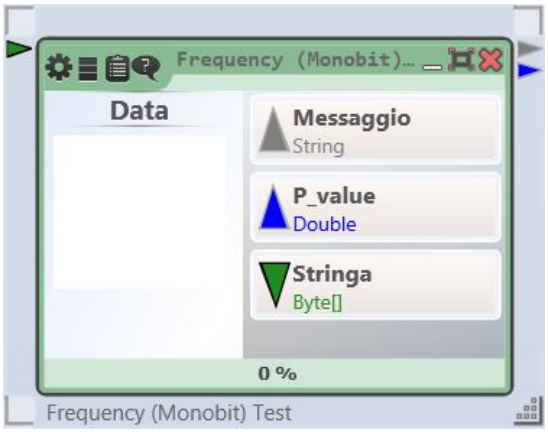

Figura 6.2 Plugin Frequency-Test

trascinando l'angolo inferiore destro o spostati cliccando e trascinando uno dei tre quadratini che si trovano agli altri vertici del rettangolo. Come si può vedere vi sono tre piccoli triangoli, uno a sinistra del rettangolo e due a destra. Il rettangolo a sinistra ha la punta rivolta verso l'interno mentre gli altri due hanno la punta verso l'esterno. Questi triangoli rappresentano rispettivamente i dati in ingresso e quelli in uscita. Nella visualizzazione Data si può vedere anche il tipo di dati utilizzati (in questo caso il dato Messaggio è di tipo stringa mentre P-*value* è di tipo Double e Stringa è un vettore di Byte). E' attraverso questi triangoli che i vari plugin possono essere collegati fra loro in un progetto. Per collegare due plugin basta cliccare sul triangolino e trascinare. In questo modo compare una linea di dimensioni variabili che deve terminare in un triangolino di un altro plugin. Il programma pre-

vede anche il controllo dei tipi poichè non è possibile collegare dati di tipo non compatibile. Per creare un progetto occorre selezionare dalla schermata di avvio nella barra dei menu il comando New e poi inserire i plugin nella finestra centrale.

Ho effettuato un certo numero di prove utilizzando i plugin per verificare il loro funzionamento analizzando alcune stringhe di lunghezza superiore a 100 bit e altre di lunghezza inferiore proprio per sottolineare l'importanza della lunghezza in questo tipo di test. Alcune delle stringhe analizzate sono le seguenti:

A= 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0;

B= 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0;

C= 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0;

D= 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0,

0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0;

E= 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;

F= 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;

G= 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;

H= 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1;

L= 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1;

M= 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1;

N= 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, , 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0;

O= 1, 0, 1, 1, 0, 1, 0, 1, 0, 1;

P= 1, 1, 1, 1, 1, 1, 1, 1, 0, 1;

Q= 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1;

R= 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1;

S= 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, , 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0;

I risultati sono mostrati in tabella:

Stringa	Frequency-Test	BlockFrequency-Test	Run-Test	LongestRun-test
A	Casuale P=0,10959858339911	Casuale (M=8) P=0,939164687614306	Casuale P=0,50079791788709	corta
B	Casuale P=0,48119168452796	Non casuale (M=2) P=0,00448265656557319	Non casuale P=2,14351805166219E-08	corta
C	Casuale P=0,23013934044341	Casuale (M=38) P=0,34901807093132	Non casuale P=0,00984664137930869	corta
D	Casuale P=0,157299207050285	Casuale (M=10) P=0,998238112131815	Non casuale P=0,00101784682539784	Non casuale P=0,0013413087746732
E	Non casuale P=4,13703174651381E-37	CNon casuale (M=5) P=3,76489357600153E-23	Non casuale P=7,49795142340932E-34	Non casuale P=6,51033482718997E-16
F	Non casuale P=2,71241132943665E-28	Non casuale (M=10) P=1,82633355380788E-22	Non casuale P=2,14002473671279E-18	Non casuale P=3,71820500312358E-14
G	Non casuale P=5,50257375069199E-25	Non casuale (M=2) P=8,69493655902833E-05	Non casuale P=1,652400325358E-14	Non casuale P=3,71820500312358E-14
H	Casuale P=0,330390048848794	Casuale (M=2) P=0,952945797586622	Non casuale P=0,000492145471815823	corta
I	Casuale P=0,0231409313087437	Casuale (M=3) P=0,445679641364612	Casuale P=0,0856283688704864	corta
L	Casuale P=0,330390048848794	Casuale (M=3) P=0,983436391519386	Casuale P=0,0312368475141128	corta
M	Casuale P=0,0138230238203767	Casuale (M=8) P=0,873728730032672	Non casuale P=0,00110974819785864	corta
N	Casuale P=0,527089256865538	Casuale (M=6) P=0,414216178242525	Casuale P=0,00565759781506437	corta
O	Casuale P=0,0114120363860017	Casuale (M=3) P=0,096472322379628	Casuale P=0,0350149810196624	corta
P	Casuale P=0,0114120363860017	Non casuale (M=7) P=0,00815097159350271	Casuale P=0,0350149810196624	corta
Q	Casuale P=0,330390048848794	Casuale (M=7) P=0,868537810877897	Non Casuale P=0,000492145471815823	corta
R	Casuale P=0,051575863620877	Casuale (M=5) P=0,382015231860716	Casuale P=0,00443955094012244	corta
S	Casuale P=0,0202750033611269	Casuale (M=6) P=0,59115369535393	Non casuale P=0,000455336057646758	corta

6.5 Il Wizard

Se nella schermata iniziale del progetto, nell'area chiamata Main Function selezioniamo l'icona che rappresenta una bacchetta magica avviamo il wizard che consente di provare le funzionalità di CrypTool2. Possiamo scegliere tra alcuni task: Encription/Decription, Cryptanalysis, Hash Functions, Mathematical Functions e Tools. Selezionando Encrryption/Decryption è possible cifrare un testo in chiaro oppure decifrare un testo cifrato. E' possibile scegliere diversi tipi di algoritmi di cifratura sia classici (per esempio Cesare, Vigenère, Sostituzine, Enigma) che moderni (a chiave simmetrica o asimme-

trica). Selezionando Cryptanalysis è possibile analizzare un testo cifrato con un algoritmo specifico e/o la chiave. Occorre inserire l'algoritmo utilizzato per la cifratura. Selezionando Hash Function è possibile ottenere la funzione di hash di una stringa inserita, calcolata secondo un algoritmo a scelta come ad esempio MD5, SHA-1, Tiger. Selezionando Mathematical Functions è possibile ottenere per esempio la fattorizzazione in numeri primi di un numero o generare un numero primo con un certo numero di cifre o eseguire un test di primalità.

Date le molteplici attività consentite, questo wizard potrebbe essere utilizzato efficacemente in classe in tutti gli ordini di scuola. Poiché attualmente insegno in una scuola media, ho pensato a qualche cosa di molto semplice che possa comunque fornire alcuni spunti per approfondimenti successivi o semplicemente per stimolare la curiosità dei miei piccoli studenti. Lo scopo dell'attività potrebbe essere quello di introdurre in modo semplice i concetti di cifratura, decifrazione e di attacco di forza bruta a uno schema di cifratura. Data l'età dei ragazzi è opportuno utilizzare un cifrario molto semplice, per esempio il cifrario di Cesare. Dopo una breve introduzione storica, si spiega l'algoritmo di cifratura e lo si fa utilizzare ai ragazzi per cifrare un piccolo testo. Poiché la funzione di crittoanalisi di CrypTool2 è basata sull'analisi delle frequenze e permette di scegliere tra inglese, francese, tedesco e spagnolo, ma non italiano, occorre scegliere un testo in una di queste lingue. Per esempio ho pensato alle righe iniziali di "Alice nel paese delle meraviglie":

> Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation

che i ragazzi cifrerebbero come segue

> "Dolfh zdv ehjlqqlqj wr jhw yhub wluhg ri vlwwlqj eb khu vlvwhu rq wkh edqn, dqg ri kdylqj qrwklqj wr gr: rqfh ru wzlfh

vkh kdg shhshg lqwr wkh errn khu vlvwhu zdv uhdglqj, exw lw kdg qr slfwxuhv ru frqyhuvdwlrqv lq lw, 'dqg zkdw lv wkh xvh ri d errn,' wkrxjkw Dolfh 'zlwkrxw slfwxuhv ru frqyhuvdwlrq"

Successivamente si può far decifrare ai ragazzi il testo cifrato, questa volta utilizzando l'apposita funzione di CrypTool2 Il testo viene decifrato correttamente. Il programma riporta anche il grafico con le frequenze delle lettere con le relative percentuali come si può vedere dalla figura seguente:

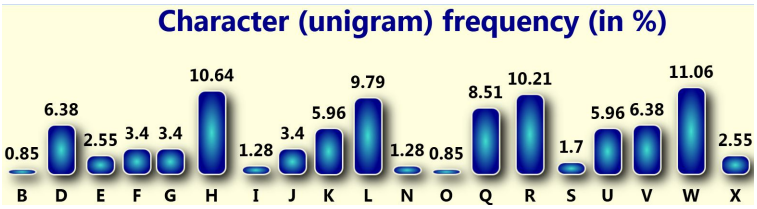

Figura 6.3 Frequenze

Si potrebbe poi far fare ulteriori prove con testi sufficientemente lunghi e mostrare come si riesca ad ottenere praticamente sempre il risultato corretto. I grafici ottenuti potrebbero essere confrontati per cercare di far loro intuire su cosa si basa l'analisi delle frequenze utilizzata dal programma. Potrebbe anche essere un utile esercizio far costruire gli stessi grafici ai ragazzi a mano o utilizzando un foglio elettronico e far ricavare loro le frequenze. Infine occorre mostrare loro i grafici delle frequenze della lingua inglese e spiegare il funzionamento di questo tipo di attacco.

Al termine del percorso e per rafforzare la loro consapevolezza del metodo è importante mostrare che per "forzare" il cifrario in questo modo, è necessario che il testo cifrato sia abbastanza lungo altrimenti l'analisi delle frequenze non ha elementi sufficienti per poter dare risultati corretti. Come esempio pratico si potrebbe ora far cifrare ai ragazzi un testo un po' più corto:

'Well!' thought Alice to herself, 'after such a fall as this, I shall think nothing of tumbling down stairs! How brave they'll all think me at home

che diventa

> 'Zhoo!' wkrxjkw Dolfh wr khuvhoi, 'diwhu vxfk d idoo dv wklv, L vkdoo wklqn qrwklqj ri wxpeolqj grzq vwdluv! Krz eudyh wkhb'oo doo wklqn ph dw krph"

e tentare di farlo decifrare a Cryptool2. Questa volta il testo decifrato è il seguente e non ha nulla a che vedere con il testo in chiaro: si ottiene infatti:

> 'Pxee!' mahnzam Tebvx mh axklxey, 'tymxk lnva t ytee tl mabl, B latee mabgd ghmabgz hy mnfuebgz whpg lmtbkl! Ahp uktox maxr'ee tee mabgd fx tm ahfx

Per qualche ragazzo il risultato sarà inaspettato e rimarrà deluso sostenendo che il programma non funziona bene o che c'è qualche trucco, qualcun altro riuscirà invece a comprendere da solo la correlazione tra la lunghezza del testo e la correttezza del metodo delle analisi delle frequenze.

6.6 Osservazioni

In questa tesi ho cercato di presentare alcuni aspetti della crittografia che mi hanno particolarmente interessato: nel primo capitolo ho mostrato alcuni momenti significativi dell'evoluzione della crittografia nella storia, nel capitolo 2 è stata sottolineato l'importanza della formulazione esatta di ipotesi e la necessità di prove rigorose di sicurezza, nel capitolo 3 è stato illustrato il cifrario one-time pad attraverso definizioni matematiche e provato l'inviolabilità attraverso una dimostrazione matematica, nel capitolo 5, attraverso procedimenti statistici (basati su teoremi) ho presentato metodi di verifica della bontà di un generatore di numeri pseudo casuali. Infine, ricollegandomi alla mia esperienza lavorativa, ho mostrato seppur brevemente, un possibile laboratorio didattico per mostrare alcuni aspetti della crittografia e per utilizzarla per superare la tradizionale didattica della Matematica che privilegia l'ascolto passivo a favore di un lavoro "per problemi" che potrebbe permettere di far toccare con mano agli studenti il modo di procedere tipico di chi fa Matematica e avvicinarli quindi all'idea che questa sia una disciplina

e una risorsa fondamentale del proprio bagaglio culturale. Concludo citando a questo proposito le parole di Neal Koblitz Professore di Matematica all'Università di Washington :

> "Cryptography has a tremendous potential to enrich math education. In the first place, it puts mathematics in a dramatic setting. [...] In the second place, cryptography provides a natural way to get students to discover certain key mathematical concepts and techniques on their own. Too often math teachers present everything on a silver platter, thereby depriving the children of the joy of discovery. In contrast, if after many hours the youngsters finally develop a method to break a cryptosystem, then they will be more likely to appreciate the power and beauty of the mathematics that they have uncovered. [...] In the third place, a central theme in cryptography is what we do not know or cannot do. The security of a cryptosystem often rests on our inability to efficiently solve a problem in algebra, number theory, or combinatorics. Thus, cryptography provides a way to counterbalance the impression that students often have that with the right formula and a good computer any math problem can be quickly solved. [...] Finally, cryptography provides an excellent opportunity for interdisciplinary projects.

Bibliografia

[1] Jonathan Katz Yehuda Lindell. *Introduction to modern cryptograpy.* Chapman & Hall/CRC, 2008.

[2] Rukhin A. Soto J. Nechvatal J. Smid M. Barker E. Leigh S. Levenson M. Vangel M. Banks D. Heckert A. Dray J. Vo S. A statistical test suite for random and pseudorandom number generators for cryptographic applications. *Special Publication 800-22 Revision 1a*, 2010.

[3] Fabrizio Luccio e Paolo Ferragina. *Crittografia. Principi Algoritmi Applicazioni.* Bollati Boringhieri, 2001.

[4] Simon Singh. *Codici e Segreti.* BUR Saggi, 2001.

[5] Neal Koblitz. Cryptography as a teaching tool. *Cryptologia*, 1997.

[6] Andrea Sgarro. *Crittografia.* Franco Muzzio Editore, 1993.

[7] Bradley L. Jones. *C#.* Apogeo, 2002.