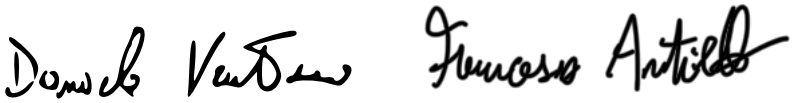
SAPIENZA
UNIVERSITÀ DI ROMA

When Randomness Comes Without a Seed — Analisi e Prospettive dei PRNG seedless

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Laurea in Informatica

Francesco Anatriello
Matricola 1957447

Relatore
Prof. Daniele Venturi



Anno Accademico 2024/2025

***When Randomness Comes Without a Seed* —**
Analisi e Prospettive dei PRNG seedless
Tesi di Laurea. Sapienza Università di Roma

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Email dell'autore: anatriello.1957447@studenti.uniroma1.it

Discovery is the ability to be puzzled by simple things.
Noam Chomsky

If you optimize everything, you will always be unhappy.
Donald Knuth

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.
John von Neumann

Sommario

L'interesse per la generazione di numeri casuali affonda le sue radici nell'antichità. L'uso dei dadi, ad esempio, è attestato già intorno a cinquemila anni fa in Mesopotamia, in India e in Cina: piccoli oggetti con facce numerate tramite incisioni o puntini, simili a quelli che ancora oggi conosciamo. Alcuni reperti sono stati rinvenuti anche nell'area mediterranea, in particolare in Egitto, e testimoniano una sorprendente continuità nell'impiego di strumenti per produrre esiti imprevedibili. In epoca romana, un metodo ancora più semplice per generare un risultato binario era il lancio della moneta, noto con l'espressione *navia aut caput* ("nave o testa"), a indicare i due simboli raffigurati sulle facce.

In questi contesti, tuttavia, la casualità non era intesa nel senso moderno del termine. L'esito di un dado o di una moneta non veniva considerato un evento privo di cause, bensì un segnale proveniente dal divino, utile a prendere decisioni o a predire il futuro. In un certo senso, già allora si riconosceva che questi strumenti producevano esiti imprevedibili, ma la loro imprevedibilità era attribuita a una volontà esterna, piuttosto che a un concetto autonomo di "randomness".

Con l'avanzare dei secoli e lo sviluppo della matematica, questa visione cambiò gradualmente. Un momento di svolta fu rappresentato dalla nascita della teoria delle probabilità tra XVII e XVIII secolo, che fornì un linguaggio formale per trattare l'incertezza. Tuttavia, la necessità concreta di numeri casuali per applicazioni scientifiche emerse con forza solo a cavallo tra XIX e XX secolo, parallelamente all'affermazione della statistica come disciplina autonoma.

Un contributo significativo arrivò da Francis Galton, scienziato britannico che nel 1890 descrisse metodi per campionare distribuzioni di probabilità utilizzando dadi. Galton, oltre a essere un pioniere della statistica, fu tra i primi a riconoscere l'utilità di sequenze casuali generate artificialmente, non solo come curiosità matematica, ma come strumenti concreti per esperimenti ed elaborazioni numeriche. La sua intuizione aprì la strada a un filone di ricerca che, nel giro di pochi decenni, avrebbe portato alla costruzione delle prime tabelle sistematiche di numeri casuali [20].

Con l'inizio del XX secolo, infatti, l'uso della statistica sperimentale e dei metodi probabilistici cominciò a richiedere grandi quantità di numeri casuali. Per soddisfare questa necessità, i ricercatori iniziarono a costruire le prime tabelle di numeri casuali, ossia elenchi stampati di cifre generati con metodi più o meno ingegnosi. Un punto di partenza fu rappresentato dal lavoro di Tippett (1927), che pubblicò una tabella contenente oltre 41.000 cifre estratte da un rapporto del censimento del 1925 [32]. Sebbene non fossero numeri "veramente casuali", il loro utilizzo rispondeva all'esigenza di avere sequenze numeriche che apparissero prive di schemi riconoscibili e che potessero essere condivise da ricercatori diversi.

Negli anni Trenta, ulteriori contributi arrivarono da Fisher e Yates (1938), che produssero una tabella di cifre selezionate da calcoli logaritmici a venti cifre decimali [13]. Il loro lavoro ebbe ampia diffusione perché includeva anche descrizioni di metodi di campionamento basati su queste sequenze, dimostrando concretamente come i numeri casuali potessero entrare nella pratica statistica.

Un ruolo cruciale fu svolto da Maurice Kendall e Bernard Babington-Smith (1938–39). I due studiosi non solo compilarono nuove tabelle, ma si preoccuparono

anche di affrontare un tema fondamentale: come valutare la "qualità" della casualità. Introdussero così una serie di test statistici per verificare le proprietà delle sequenze finite: il test di frequenza, per controllare che ogni cifra compaia con la stessa probabilità teorica; il test seriale, per valutare la distribuzione delle coppie di cifre consecutive; il *poker test*, basato sulla frequenza di particolari combinazioni di cinque cifre; e il *gap test*, che misura le distanze tra due ricorrenze della stessa cifra [16].

Questi strumenti consentirono di distinguere tra sequenze che "sembravano" casuali e sequenze che lo erano davvero in senso statistico. In altre parole, segnarono il passaggio da un uso puramente pratico dei numeri casuali a un'analisi metodologica sulla loro qualità. Kendall e Babington-Smith sperimentarono anche forme di generazione semi-automatica, come un disco di cartone numerato da 0 a 9 fatto ruotare a velocità variabile, la cui cifra visibile veniva scelta da una lampada che lampeggiava a intervalli irregolari. Sebbene lento e non completamente automatizzato, questo dispositivo può essere considerato uno dei primi tentativi di generazione meccanica di numeri casuali.

Il vero salto di qualità nella produzione di numeri casuali si ebbe però nel secondo dopoguerra, quando l'avvento dell'elettronica rese possibile pensare a dispositivi automatici più sofisticati. Nel 1947 la RAND Corporation, un istituto di ricerca statunitense, avviò un progetto ambizioso: costruire una tabella di un milione di cifre casuali da mettere a disposizione di statistici, matematici e fisici.

Per ottenere questa enorme sequenza, i ricercatori svilupparono un dispositivo elettronico capace di emettere impulsi a intervalli casuali, con una frequenza di circa 100.000 al secondo. Questi impulsi venivano contati mediante un contatore a 5 bit, in grado di produrre 32 possibili valori. Per ricondurli al sistema decimale, venti di questi valori venivano mappati sulle cifre da 0 a 9, mentre i rimanenti erano scartati. Le cifre venivano poi registrate su schede perforate IBM, pronte per essere utilizzate nei calcolatori dell'epoca.

I primi test statistici mostrarono alcune imperfezioni: in porzioni consistenti della sequenza, le cifre dispari risultavano leggermente più frequenti delle pari. Per correggere questo difetto, fu applicata una tecnica di "randomizzazione composta": ogni cifra veniva trasformata sommando (modulo 10) il suo valore a quello della cifra corrispondente nella scheda precedente. Questo accorgimento permise di ridurre il bias e ottenere una distribuzione più equilibrata.

Il lavoro culminò nel 1955 con la pubblicazione del volume *A Million Random Digits with 100,000 Normal Deviates* [29], che divenne rapidamente un punto di riferimento per la ricerca. Questo singolare libro, composto da pagine di sequenze numeriche, rappresentò per anni la risorsa principale per chiunque avesse bisogno di numeri casuali affidabili.

Il progetto RAND segnò quindi una svolta: per la prima volta la generazione casuale veniva affrontata con criteri scientifici e strumenti elettronici automatizzati. Tuttavia, anche questo approccio presentava limiti pratici. Consultare libri di cifre o gestire migliaia di schede perforate era un'operazione lenta e poco adatta alle esigenze crescenti della simulazione numerica. Proprio questa difficoltà spinse i ricercatori a cercare soluzioni alternative: non più estrarre numeri da archivi già pronti, ma produrli direttamente attraverso procedure algoritmiche integrate nei computer stessi.

Nacquero così, tra la fine degli anni Quaranta e l'inizio degli anni Sessanta, i primi *Pseudo Random Number Generators* (PRNGs), algoritmi matematici capaci di produrre sequenze di numeri solo apparentemente casuali. Uno dei più celebri fu il *Linear Congruential Generator* (LCG) proposto da D.H. Lehmer nel 1949 [21]: un metodo semplice, ma efficace, che gettò le basi per decenni di ricerca nel campo.

Da questo momento in avanti, l'evoluzione della generazione casuale si sposta definitivamente nell'ambito dell'informatica e della teoria degli algoritmi. Per comprenderne appieno la portata, è però necessario chiarire che cosa si intenda con RNG e quali caratteristiche fondamentali lo definiscano; è ciò di cui ci occuperemo nei prossimi capitoli.

Indice

Capitolo 1

Random Number Generators: Fondamenti e Motivazioni

1.1 Definizione di RNG e modelli principali

Un *Random Number Generator* (**RNG**) è un meccanismo, di natura fisica o algoritmica, progettato per produrre sequenze che si comportino come casuali. Come anticipato nell'introduzione, gli RNG hanno molteplici applicazioni; in questo lavoro, tuttavia, l'attenzione sarà rivolta esclusivamente al loro impiego in ambito **crittografico**.

Una prima distinzione riguarda i *True RNG* (**TRNG**) e i *Pseudo-RNG* (**PRNG**). I primi si basano su sorgenti fisiche intrinsecamente aleatorie, come il rumore termico nei circuiti elettronici, il decadimento radioattivo o fenomeni quantistici ottici. Il vantaggio è che i valori prodotti non sono deterministici, ma lo svantaggio risiede nella velocità limitata e nella possibile presenza di bias che devono essere corretti con algoritmi di post-processing.

I generatori algoritmici, invece, producono sequenze a partire da uno stato interno inizializzato da un seme. Tra i primi esempi si ricorda il metodo dei *quadrati centrali* di John von Neumann [33], in cui si elevava al quadrato il seme e si prendevano le cifre centrali come nuova uscita. L'idea, pionieristica, soffriva di periodi estremamente brevi e distribuzioni non uniformi. Più solida si rivelò la classe dei *Linear Congruential Generators* (LCG) [21], definiti dalla ricorrenza

$$X_{n+1} = (aX_n + c) \bmod m,$$

dove i parametri a, c e m determinano la qualità e la lunghezza del periodo. Questo approccio, semplice e di facile implementazione, divenne a lungo lo standard nelle simulazioni numeriche.

Sviluppi successivi hanno portato a generatori più avanzati, tra i più famosi ricordiamo il *Mersenne Twister* [23], celebre per il periodo eccezionalmente lungo $(2^{19937} - 1)$ e per la proprietà di equidistribuzione fino a 623 dimensioni, gli *Xorshift* [22], introdotti da George Marsaglia e basati su operazioni bitwise (XOR e shift) per ottenere sequenze leggere e veloci con buone proprietà statistiche, e il *Permuted Congruential Generator* (PCG), proposto da O'Neill nel 2014 [28], che modernizza

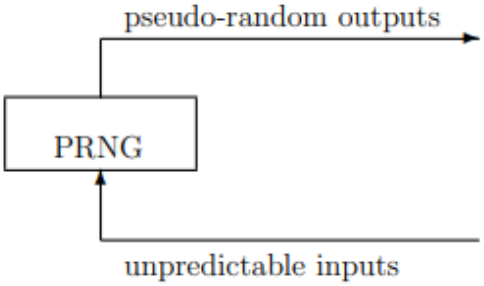


Figura 1.1. Black-box view di un PRNG [19]

gli LCG introducendo permutazioni non lineari per migliorarne la qualità statistica mantenendo semplicità ed efficienza d'uso.

Questi generatori algoritmici hanno il vantaggio di essere veloci ed efficienti, ma la loro natura deterministica li rende inadatti a garantire da soli sicurezza crittografica. Per questo motivo sono stati sviluppati i *Cryptographically Secure PRNG* (**CSPRNG**), basati su primitive consolidate come cifrari a blocchi, funzioni hash e cifrari a flusso, e regolati da standard internazionali quali la famiglia `NIST SP 800-90` [26]. Un esempio rilevante è *Fortuna*, proposto da Schneier e Ferguson [30], che adotta una struttura a più pool di entropia e l'impiego di primitive crittografiche robuste per garantire sicurezza anche in scenari di compromissione parziale.

Nel modello matematico, un RNG può essere rappresentato come una quintupla (S, μ, f, U, g) [19]. Qui:

- S è un insieme finito di stati,
- μ è una distribuzione di probabilità su S, usata per selezionare lo stato iniziale s_0,
- $f : S \to S$ è la funzione di transizione che aggiorna lo stato a ogni passo,
- U è lo spazio dei valori di uscita (tipicamente l'intervallo $(0, 1)$),
- $g : S \to U$ è la funzione di output.

L'evoluzione del generatore è descritta dalla ricorrenza

$$s_i = f(s_{i-1}), \quad i \geq 1,$$

e l'uscita associata allo stato s_i è

$$u_i = g(s_i) \in U.$$

La sequenza $(u_0, u_1, u_2, \ldots)$ costituisce i numeri casuali prodotti dal generatore. Questo modello, pur astratto, è particolarmente utile perché permette di analizzare con rigore le proprietà fondamentali di un RNG, come il periodo, l'uniformità e l'indipendenza, oggetto del paragrafo successivo.

1.2 Proprietà desiderabili di un RNG

Dopo aver chiarito che cosa si intenda con Random Number Generator, è necessario stabilire quali proprietà definiscono la "qualità" di un generatore. Non tutti gli RNG sono equivalenti: alcuni producono sequenze con caratteristiche matematiche soddisfacenti ma poco utili in crittografia, altri invece offrono garanzie di sicurezza molto più robuste. In letteratura, sono stati individuati diversi criteri fondamentali [17, 19, 24, 26].

Proprietà statistiche Tra i requisiti di base si annoverano: un **periodo sufficientemente lungo**, l'**efficienza** in termini di velocità e memoria, la **riproducibilità** (stesso seme $\rightarrow$ stessa sequenza) e la **portabilità** su piattaforme diverse [19]. In scenari come le simulazioni parallele, risulta utile anche la possibilità di generare **flussi indipendenti** tramite tecniche di *jump-ahead* dello stato.

Un RNG deve inoltre garantire valori **uniformemente distribuiti** e privi di correlazioni statistiche. Non si tratta solo di uniformità a livello di singolo valore, ma anche di **uniformità multidimensionale**: i punti devono distribuirsi in maniera equilibrata nello spazio t-dimensionale $(0, 1)^t$. Questo aspetto viene spesso valutato tramite la **discrepancy**, che misura la distanza fra la distribuzione empirica e quella ideale [17, 19].

Accanto ai criteri teorici, un RNG deve essere validato tramite **test statistici empirici**. L'ipotesi nulla H_0 assume che la sequenza sia indipendente e identicamente distribuita ($U(0, 1)$). Test classici includono il serial test, il poker test, il chi-square e il gap test. Nessun RNG a stati finiti può superare ogni possibile test: si distinguono quindi generatori "deboli", che falliscono test semplici, da generatori "robusti", che falliscono solo test particolarmente complessi. Tra le batterie più note vi sono `DIEHARD`, `NIST SP 800-22` e `TestU01` [19, 25].

Proprietà di sicurezza In ambito crittografico, le proprietà statistiche non sono sufficienti. È richiesta **imprevedibilità computazionale**: un RNG è considerato sicuro se nessun algoritmo efficiente può distinguere le sue uscite da sequenze veramente casuali, né predire i valori futuri conoscendo quelli passati. Questa definizione, vicina al concetto di *polynomial-time perfect RNG*, è alla base dei **CSPRNG** [24].

Secondo gli standard NIST [26], un CSPRNG deve inoltre garantire proprietà di:

- **forward security**, ovvero la resistenza al recupero di uscite passate nel caso in cui lo stato venga compromesso;
- **backward security**, cioè l'impossibilità per un avversario di predire le uscite future conoscendo lo stato corrente.

1.3 Il problema del seeding tradizionale

Come già discusso nei paragrafi precedenti, la sicurezza di un PRNG dipende in modo cruciale dal seme iniziale. La qualità di questo seme è legata alla quantità di *entropia* che contiene, ossia all'imprevedibilità della sorgente che lo ha generato. In termini intuitivi, l'entropia misura quanta incertezza è associata a una variabile casuale: maggiore è l'entropia, più difficile sarà per un avversario prevederne il valore. Formalmente, data una variabile casuale X con distribuzione $p(x)$, l'entropia di Shannon si definisce come

$$H(X) = -\sum_{x} p(x)\, \log p(x),$$

che rappresenta il numero medio di bit necessari per descrivere l'incertezza di X.

In questo quadro, Dodis et al. [7] osservano che i modelli teorici tradizionali, pur variando nei dettagli, condividono due principi fondamentali:

1. Il PRNG dovrebbe funzionare anche in presenza di sorgenti di entropia avversarie, purché tali sorgenti forniscano, prima o poi, una quantità sufficiente di entropia (tali sorgenti vengono dette "legittime").

2. Assumere più struttura oltre alla sola entropia è indesiderabile e fragile, poiché ciò richiede una comprensione piuttosto dettagliata delle proprie sorgenti di entropia.

Queste assunzioni, se da un lato garantiscono grande generalità, dall'altro comportano un caveat significativo: l'estrazione di casualità non può essere deterministica, ed è quindi necessario ricorrere a *seeded extractors*. Di conseguenza, i modelli tradizionali finiscono per dipendere da un seme iniziale, che però introduce fragilità teoriche e pratiche.

Vengono sottolineati vari scenari critici:

- se la motivazione per cui si usa un PRNG è proprio la mancanza di entropia, non è affatto scontato riuscire a generare un seme realmente casuale;

- anche quando il seme è uniforme, è essenziale che le fonti di entropia rimangano indipendenti dal seme stesso, un'ipotesi spesso fragile nei sistemi reali;

- un avversario può ottenere informazioni sul seme, sia direttamente (compromissione della memoria) sia indirettamente, tramite output prodotti in condizioni di bassa entropia — fenomeno noto come *premature next attack*;

- l'uso di un seme fissato una volta per tutte, pratica comune in implementazioni reali, elimina qualsiasi garanzia di sicurezza;

- persino la combinazione di più fonti di entropia può risultare rischiosa: se anche una sola dipende dal seme, le garanzie teoriche decadono.

1.4 Perché i Seedless PRNGs?

Qua arriviamo al cuore di questa tesi. Le criticità appena discusse mostrano che la dipendenza da un seme iniziale non è un dettaglio secondario, ma un limite strutturale dei PRNG tradizionali. In altre parole, anche assumendo generatori algoritmici robusti, le garanzie teoriche e pratiche possono crollare se il seme è scarso, correlato o compromesso.

Ci troviamo dunque di fronte a un vero e proprio dilemma: da un lato, vorremmo supportare sorgenti di entropia generali, per le quali l'estrazione seedless è impossibile; dall'altro, l'estrazione con seme è possibile solo sotto ipotesi di indipendenza molto forti, pericolose e difficili da garantire nella pratica, e che quindi si preferirebbe evitare.

L'obiettivo dei **Seedless PRNGs** è fornire una soluzione significativa a questo dilemma: progettare generatori che rimangano seedless pur rispettando i due principi fondamentali menzionati in precedenza, ossia la capacità di operare anche con sorgenti di entropia avversarie ma "legittime", e l'assenza di assunzioni strutturali aggiuntive sulle sorgenti oltre alla mera quantità di entropia.

Questa prospettiva, al tempo stesso teorica e pratica, rappresenta un cambio di paradigma rispetto ai modelli seeded. Nel prossimo capitolo analizzeremo nel dettaglio le costruzioni seedless proposte da Dodis et al. [7], basate su primitive crittografiche come le funzioni hash, e vedremo come queste possano fornire garanzie di sicurezza formali in scenari dove i PRNG tradizionali falliscono.

Capitolo 2

Il modello Seedless PRNG

2.1 Dal PRNG classico al paradigma Seedless

Il modello introdotto da *Coretti, Dodis, Karthikeyan e Tessaro* [7] costituisce un tentativo organico di fornire una base formale alla generazione pseudocasuale priva di seme, impiegando le **funzioni di hash crittografiche (CHFs)** come principale strumento teorico. L'obiettivo è inquadrare in modo rigoroso un principio già consolidato nella pratica: l'uso delle funzioni hash come elemento centrale dei PRNG moderni.

L'analisi infatti parte da un dato ormai evidente. Tutti i generatori software di uso generale, così come le raccomandazioni presenti negli standard attuali, si fondano sull'impiego di CHFs. Nonostante ciò, mancava una giustificazione teorica capace di spiegare perché tale dipendenza fosse non solo conveniente, ma necessaria per la sicurezza del sistema, specialmente in assenza di un seme indipendente.

Il punto chiave è rappresentato dalla funzione di aggiornamento del PRNG, detta *refresh*. Dato uno stato interno S e un nuovo input entropico X, essa produce un nuovo stato $S' = \text{refresh}(S, X)$, con lo scopo di assorbire l'entropia di X e accrescerne la quantità all'interno dello stato. In termini teorici, questa operazione corrisponde alla **condensazione dell'entropia** (*entropy condensing*).

Quando è disponibile un seme indipendente, tale operazione è relativamente semplice: si può costruire un *condenser* lineare, del tipo

$$S' = aS + X.$$

dove a funge da parametro casuale separato. Nel modello *seedless*, invece, questa indipendenza non è più garantita: la sorgente X può risultare parzialmente correlata allo stato S o ai parametri interni del sistema, compromettendo la distinzione fra entropia "nuova" e "accumulata". In tali condizioni, la funzione di aggiornamento non solo perde le proprie garanzie formali di sicurezza, ma può produrre **uscite completamente prive di entropia**, anche a partire da input apparentemente "ad alta casualità".

Per superare tale fragilità, è necessario costruire **condensatori di entropia crittograficamente forti**, ossia funzioni che non dipendano dall'indipendenza statistica delle sorgenti, ma da proprietà come la **resistenza alle collisioni** e la **resistenza alla preimmagine**.

Le CHFs soddisfano intrinsecamente questi requisiti, rappresentando il candidato naturale per definire una fase di *refresh* sicura in contesti privi di semi segreti.

Dalla sicurezza computazionale alla nuova nozione di entropia. In questa prospettiva, la sicurezza del PRNG non deriva più dall'isolamento di un seme iniziale, ma dalle caratteristiche strutturali delle primitive hash. Tali funzioni vengono quindi analizzate in modelli idealizzati — come il *random oracle*, l'*ideal cipher* e la *random permutation* — che consentono di dimostrare formalmente le proprietà di condensazione e di accumulo dell'entropia. Il ricorso a tali modelli non è un mero artificio analitico: serve a formalizzare l'idea che le CHFs agiscano come estrattori di entropia anche in presenza di input correlati o parzialmente avversari.

Tuttavia, l'impiego di funzioni hash come base per la generazione seedless solleva una questione più profonda, di natura teorica. La misura classica di entropia, $H_\infty(X)$, quantifica la dispersione statistica dei valori di una sorgente indipendente, ma non riesce a cogliere le dipendenze computazionali introdotte dallo stato interno o dall'oracolo. Nei PRNG seedless, tali dipendenze sono inevitabili: gli input successivi possono essere influenzati dalle interazioni precedenti e dalle risposte dell'oracolo stesso. Diventa quindi necessario introdurre una forma di entropia "relativa", che esprima quanta imprevedibilità rimanga nella sorgente dopo aver tenuto conto delle informazioni osservabili dal sistema.

La sezione successiva svilupperà questa nuova nozione di entropia condizionata, mostrando come essa permetta di distinguere fra sorgenti di entropia naturali e manipolate, e di definire formalmente le condizioni in cui una fonte può essere considerata *legittima*. Ciò permetterà di comprendere come l'accumulo di entropia e la sicurezza del generatore possano essere garantiti anche in assenza di un seme indipendente.

2.2 Concetto di entropia e modellazione delle fonti oracle-dipendenti

La ridefinizione dell'entropia nel modello seedless nasce dalla constatazione di un limite teorico fondamentale, ovvero l'estrazione di casualità priva di seme è, in generale, impossibile per sorgenti di entropia arbitrarie.

Qualsiasi estrattore deterministico G può essere vulnerabile a un cosiddetto **extractor-fixing attack**: un avversario campiona ripetutamente un input casuale X finché il primo bit dell'uscita $G(X)$ assume un valore desiderato (ad esempio 0). La distribuzione di X risultante possiede un'elevata entropia in senso classico, ma l'output $G(X)$ è evidentemente non uniforme. Questo mostra come, nei modelli puramente informazionali, una sorgente possa apparire ad alta entropia pur producendo risultati del tutto prevedibili se l'avversario è in grado di influenzarne la generazione.

Come già accennato, le funzioni di hash crittografiche rappresentano l'elemento portante del paradigma seedless; nel quadro teorico che si delinea, il loro ruolo va oltre la mera implementazione pratica. Di fatto, esse diventano il **mezzo** attraverso cui il modello supera i limiti informazionali dell'estrazione, rendendo l'attacco di extractor-fixing non solo marginale, ma **l'unico** scenario di rischio possibile.

Passiamo ora ad una definizione chiave:

Definizione 2.1. La misura classica di entropia, detta **min-entropia**, quantifica l'imprevedibilità di una sorgente nel caso peggiore. È definita come

$$H_\infty(X) = -\log_2 \left(\max_x \Pr[X = x] \right),$$

e indica il numero minimo di bit effettivamente imprevedibili che una variabile casuale può fornire. L'obiettivo del modello seedless non è soltanto sostituire un'ipotesi di sicurezza con un'altra, ma costruire un quadro coerente in cui l'estrazione di casualità sia *realisticamente possibile*.

A tal fine viene introdotta una nuova definizione di min-entropia, compatibile con l'esistenza delle CHFs, che permette di distinguere formalmente due categorie di sorgenti:

(a) le sorgenti *artificiali*, ottenute intenzionalmente mediante procedure di extractor-fixing, che non vengono più considerate legittime poiché possiedono entropia trascurabile secondo la nuova definizione;

(b) le sorgenti *naturali*, come quelle utilizzate dai principali sistemi operativi o dai dispositivi fisici di generazione di casualità, che mantengono invece un buon livello di entropia secondo questa misura.

Questa distinzione rende nuovamente possibile l'estrazione seedless, purché si lavori in un contesto computazionale che rifletta le proprietà delle CHFs. In questa prospettiva, la sicurezza del generatore dipende direttamente dalle primitive hash impiegate, come SHA-2, SHA-3 e l'HKDF, che incarnano in forma concreta le proprietà di *collision-resistance*, *preimage-resistance* e di accumulo progressivo dell'entropia richieste dal modello. Tali funzioni, già ampiamente standardizzate, rappresentano oggi le migliori approssimazioni pratiche di un oracolo casuale e costituiscono la base per costruzioni seedless teoricamente fondate.

Tuttavia, la dimostrazione rigorosa di tali proprietà risulta al momento raggiungibile solo nei **modelli idealizzati di computazione**, come i già citati *random oracle*, *ideal cipher* e *random permutation model*. Questa scelta, comune in molte aree della crittografia simmetrica, consente di lavorare in un quadro formale non tautologico e di isolare i principi strutturali del paradigma seedless, attualmente non ottenibili nel modello standard, nemmeno per costrutti più semplici come i condensatori *average-case*.[1]

Il caso monolitico e la dipendenza dall'oracolo. Per chiarire operativamente la nuova nozione di entropia introdotta nel modello, si considera dapprima un caso semplificato, denominato **monolithic seedless extraction**. In questo scenario, la sorgente di entropia X viene fornita come un unico blocco di dati, mentre una funzione di hash G, modellata come un oracolo casuale monolitico (ossia un'entità teorica che

[1] I *condensatori average-case* sono funzioni che, a partire da una sorgente con entropia parziale, producono un'uscita di entropia più concentrata anche nel caso in cui la sorgente sia correlata o "rumorosa". Rappresentano una generalizzazione dei condensatori classici, ma richiedono ipotesi più forti o modelli idealizzati per garantire sicurezza formale. [8]

risponde a interrogazioni come se implementasse una funzione ideale perfettamente casuale), viene utilizzata per produrre l'uscita $R = G(X)$. Si ignorano volutamente i dettagli interni della funzione hash, concentrandosi sul legame concettuale tra sorgente e oracolo.

Se G è un oracolo casuale perfetto, agisce come un buon estrattore di entropia. Un risultato classico (ad esempio [27]) mostra infatti che, per una sorgente X con entropia minima γ^* (ovvero almeno γ^* bit effettivamente imprevedibili), la probabilità che un *distinguer* D[2] riesca a distinguere $G(X)$ da un valore realmente casuale è limitata da

$$\Pr[D(G(X)) = 1] \leq q \cdot 2^{-\gamma^*},$$

cioè, la probabilità che un distinguitore riesca a riconoscere $G(X)$ come non casuale è proporzionale al numero di query q e decresce esponenzialmente con la min-entropia γ^* della sorgente.

Questa apparente semplicità nasconde però un presupposto critico: la dimostrazione vale solo se la distribuzione della sorgente X è completamente indipendente dall'oracolo G, cioè se il modo in cui la sorgente genera i propri valori non è influenzato dalle risposte dell'oracolo. In pratica, questa ipotesi equivale a introdurre un seme di lunghezza esponenziale, che rende l'estrazione teoricamente semplice ma irrealistica. Nei sistemi reali, invece, le sorgenti di entropia sono spesso influenzate dalle stesse primitive crittografiche che ne utilizzano l'output. Per rappresentare più fedelmente tale dinamica, il modello seedless deve quindi ammettere che la sorgente X dipenda parzialmente dall'oracolo G.

Sorgenti oracle-dipendenti e nuova definizione di entropia. Il caso monolitico evidenzia la necessità di una definizione più generale. Per rappresentare in modo coerente le dipendenze reali tra sorgente e oracolo, il modello introduce il concetto di **sorgente oracle-dipendente**. La sorgente X viene trattata come parte del comportamento di un avversario $\mathcal{A}$, composto da due stadi $\mathcal{A}^G = (\mathcal{A}_1^G, \mathcal{A}_2^G)$.

Il primo stadio genera X interagendo con l'oracolo G e produce una trascrizione Σ, ossia l'insieme delle risposte ricevute dall'oracolo, e una lista di query $\mathcal{L}$ che contiene gli input inviati e il loro ordine. Tali informazioni vengono poi passate al secondo stadio, che deve distinguere l'uscita $R = G(X)$ da un valore uniforme. Affinché la definizione sia significativa, è necessario che X sia *legittima*,[3] cioè che conservi entropia sufficiente anche conoscendo la trascrizione Σ.

Nel modello standard, questa condizione si formalizza imponendo che

$$H_\infty(X \mid \Sigma) \geq \gamma^*,$$

dove $H_\infty(X \mid \Sigma)$[4] rappresenta la quantità di entropia residua di X dato lo stato informativo Σ. Tuttavia, tale vincolo risulta debole: un avversario può ancora

[2] Un distinguitore è un algoritmo probabilistico che, ricevuto un campione, tenta di determinare se esso provenga da una distribuzione realmente casuale o da un generatore pseudocasuale.

[3] Una sorgente è detta *legittima* quando conserva una quantità minima di entropia anche dopo che un avversario ha osservato tutte le sue interazioni con l'oracolo, cioè quando la sua imprevedibilità non può essere annullata da tali informazioni.

[4] L'*entropia condizionata* misura quanta imprevedibilità conserva una sorgente dopo aver considerato alcune informazioni note; in questo caso, rappresenta l'entropia residua di X una volta osservata la trascrizione Σ.

memorizzare porzioni dell'output $G(X)$ nella trascrizione, vanificando la garanzia residua. Anche rafforzando la condizione a $H_\infty(X \mid \Sigma, G) \geq \gamma^*$, l'attacco rimane possibile, poiché un campionatore adattivo, cioè un algoritmo che genera la sorgente X decidendo le proprie query all'oracolo in base alle risposte precedenti, può comunque sfruttare informazioni residue sull'oracolo.

La soluzione consiste nel condizionare la min-entropia non solo su Σ, ma anche sulla lista di query $\mathcal{L}$ effettuate durante la generazione:

$$H_\infty(X \mid (\Sigma, \mathcal{L})) \geq \gamma^*.$$

Questa formulazione impone che la sorgente mantenga almeno γ^* bit di entropia anche conoscendo tutte le query e le risposte dell'oracolo. In termini intuitivi, l'avversario non può conservare informazioni sufficienti a ridurre l'entropia residua di X sotto la soglia di sicurezza.

Dal punto di vista concettuale, la legittimità di una sorgente oracle-dipendente non dipende più dalla sua indipendenza formale, ma dal fatto che le dipendenze con l'oracolo non permettano una previsione efficiente dell'output. Ciò cattura l'essenza delle sorgenti di entropia reali, che possono dipendere dal sistema ma non essere completamente deterministiche. Su questa base, il modello mostra che è possibile realizzare un'**estrazione seedless completa** a partire da sorgenti legittime, fornendo così il fondamento teorico per le costruzioni presentate nelle sezioni successive.

2.3 Introduzione alla Seedless Extraction

Dopo aver analizzato il ruolo delle funzioni di hash come condensatori di entropia e introdotto la nozione di entropia oracle-dipendente, possiamo ora passare alla fase costruttiva del paradigma seedless. Questa parte è dedicata al meccanismo di **estrazione seedless** e alla sua estensione verso le costruzioni di **Seedless PRNGs**. L'obiettivo è mostrare come l'entropia proveniente da sorgenti parzialmente avverse possa essere accumulata e rigenerata nel tempo, fino a produrre uscite pseudocasuali indistinguibili da quelle di un generatore ideale.

Per introdurre queste costruzioni è necessario chiarire alcuni concetti preliminari, che serviranno a fissare la notazione e i presupposti teorici utilizzati per modellare, analizzare e garantire l'accumulo sicuro di entropia nel tempo. In particolare, prima di formalizzare il meccanismo di estrazione seedless, è opportuno richiamare alcune nozioni di base di teoria dell'informazione e di sicurezza crittografica. Questi concetti costituiranno gli strumenti matematici impiegati per misurare la casualità di una sorgente e per descrivere il comportamento di un avversario che interagisce con un sistema ideale.

2.3.0 Preliminari tecnici

Distanza Statistica e Probabilità di Previsione. La distanza statistica (**SD**) tra due variabili casuali X e Y, definite sullo stesso dominio, è data da:

$$\mathrm{SD}(X, Y) = \tfrac{1}{2} \sum_x \big| \Pr[X = x] - \Pr[Y = x] \big|.$$

Questa misura quantifica quanto le due distribuzioni differiscono. In termini intuitivi, rappresenta la probabilità massima con cui un osservatore ideale, dopo aver visto un campione, può stabilire se esso provenga da X o da Y. Se $\mathrm{SD}(X, Y) = 0$, le due distribuzioni sono identiche; se $\mathrm{SD}(X, Y) = 1$, sono completamente distinguibili. Nel contesto crittografico, una distanza statistica trascurabile indica che X e Y sono *indistinguibili*, cioè che nessun avversario efficiente può differenziarle con probabilità significativamente superiore al caso.

Un secondo concetto fondamentale è la **probabilità di previsione** (*prediction probability*), che misura quanto sia facile indovinare il valore assunto da una variabile casuale. Per una variabile X si definisce come:

$$\mathrm{Pred}(X) := \max_x \Pr[X = x].$$

Questo valore rappresenta la probabilità di successo dell'avversario ottimale, cioè quello che, conoscendo la distribuzione di X, sceglie sempre il valore più probabile. Nel caso di una distribuzione uniforme su n bit, $\mathrm{Pred}(X) = 2^{-n}$; invece, se X è deterministica, $\mathrm{Pred}(X) = 1$. La probabilità di previsione fornisce quindi una misura diretta dell'imprevedibilità della sorgente: più è bassa, maggiore è la casualità effettiva di X.

Quando la sorgente X è osservata in un contesto informativo, ad esempio in presenza di un'informazione ausiliaria Y (o di uno stato interno noto all'avversario), si considera la versione condizionata:

$$\mathrm{Pred}(X|Y) = \mathbb{E}_{y \leftarrow Y}\big[\max_x \Pr[X = x \mid Y = y]\big].$$

Qui l'operatore di media $\mathbb{E}_{y \leftarrow Y}$ tiene conto di tutti i possibili valori che Y può assumere, riflettendo la capacità media di un avversario ottimale di prevedere X dopo aver osservato Y. Questa quantità sarà fondamentale per definire la sicurezza degli estrattori seedless rispetto a sorgenti parzialmente correlate.

Da essa deriva la **min-entropia condizionata**, già introdotta nella Sezione 2.2, che esprime la quantità minima di imprevedibilità che una sorgente X conserva anche conoscendo Y:

$$H_\infty(X|Y) = -\log(\mathrm{Pred}(X|Y)).$$

Nel seguito, tale misura sarà impiegata per caratterizzare le sorgenti *legittime* e per formalizzare le garanzie di sicurezza dell'estrazione seedless.

Giochi di sicurezza e vantaggio dell'avversario. Le proprietà di sicurezza dei modelli seedless vengono descritte attraverso un esperimento tra un *challenger* e un avversario $\mathcal{A}$, entrambi con accesso a una primitiva ideale P (ad esempio un oracolo casuale o una cifratura perfetta). Il challenger sceglie un bit casuale $b \in \{0, 1\}$ e definisce due mondi distinti:

- il **mondo reale** ($b = 0$), in cui l'avversario interagisce con il sistema effettivo;
- il **mondo ideale** ($b = 1$), in cui le operazioni sono emulate da un oracolo perfettamente casuale.

L'obiettivo dell'avversario è indovinare il valore di b, cioè determinare se sta osservando il mondo reale o quello ideale. Il suo **vantaggio** (*advantage*) è definito come:

$$\mathrm{Adv}_{\mathcal{A}} = 2 \cdot \left| \Pr[\mathcal{A} \text{ vince}] - \tfrac{1}{2} \right| = \left| \Pr[\mathcal{A} = 1 | b = 0] - \Pr[\mathcal{A} = 1 | b = 1] \right|.$$

In altre parole, misura quanto le due probabilità di successo nei due mondi differiscono tra loro. Un sistema è tanto più sicuro quanto più questo vantaggio è trascurabile, cioè quando nessun avversario efficiente può distinguere i due scenari con probabilità significativamente superiore al caso.

2.3.1 Definizione *formale* di Seedless Extraction

Come primo passo verso la costruzione completa dei PRNG seedless, questa sezione affronta il problema più semplice ma fondamentale, quello dell'**estrazione seedless**, ossia la produzione di bit statisticamente uniformi a partire da **sorgenti deboli ma ad alta entropia**. L'estrazione rappresenta la base concettuale dell'intera architettura seedless e può essere vista come l'equivalente teorico della *post-compromise security* [5]: la capacità di recuperare casualità anche dopo una compromissione parziale dello stato interno.

Dal punto di vista formale, la sicurezza di un estrattore seedless dipende dall'entropia dell'input fornito dall'avversario, condizionata sullo stato interno e sulle interrogazioni effettuate a una primitiva ideale P. Sono considerate due nozioni di sicurezza:

- **Computazionale**, quando l'avversario dispone di risorse limitate e la primitiva P è trattata come funzione resistente alle collisioni o alla preimmagine;
- **Informazionale** (o *information-theoretic*), che garantisce sicurezza assoluta, indipendente dalla potenza di calcolo.

Nel caso informazionale, la garanzia è particolarmente forte: l'uscita dell'estrattore deve restare indistinguibile da una stringa uniforme anche se, dopo aver scelto l'input, l'avversario ottiene la tabella completa della primitiva ideale. Questa condizione, detta *everlasting security*[6], esprime la resilienza della generazione seedless anche a posteriori.

Inoltre, è importante chiarire che le nozioni appena descritte possono essere analizzate in due contesti distinti:

- Nel **modello ideale**, dove le primitive crittografiche (come le funzioni di hash o le cifrature) vengono trattate come oracoli perfettamente casuali; questa astrazione semplifica le dimostrazioni e permette di isolare le proprietà strutturali del sistema.

[5] Con *post-compromise security* si intende la capacità di un sistema crittografico di ristabilire la propria sicurezza anche dopo che una parte dello stato segreto è stata compromessa. In tale modello, il sistema deve poter "rigenerare" entropia da nuove fonti, garantendo che le uscite future risultino nuovamente indistinguibili da valori casuali. Nel contesto dei PRNG seedless, questa proprietà coincide con la possibilità di recuperare casualità a partire da sorgenti parzialmente avverse.[4]

[6] La *everlasting security* garantisce che la casualità prodotta resti indistinguibile dall'uniforme anche se, in un secondo momento, l'avversario ottiene conoscenza completa della funzione o dello stato interno; cfr. Harnik–Naor [14].

- Nel **modello standard**, dove si lavora con primitive concrete e sorgenti realistiche, valutando la sicurezza rispetto alla potenza computazionale effettiva dell'avversario.

In assenza di un seme indipendente, la sicurezza dell'estrazione dipende interamente dalla *legittimità della sorgente*, ossia dalla sua capacità di conservare una quantità minima di entropia anche dopo le interazioni con l'oracolo. Questa condizione si formalizza tramite l'entropia condizionata $H_\infty(X \mid \Sigma, \mathcal{L})$, che misura l'imprevedibilità residua di X una volta note le risposte dell'oracolo (Σ) e la lista di query ($\mathcal{L}$). Per formalizzare il modello, si considera una famiglia di primitive ideali $\mathcal{P}$ e si fissa $P \in \mathcal{P}$ scelta uniformemente a caso. Un estrattore seedless è un algoritmo

$$\mathsf{ext}^P : \mathcal{X} \to \mathcal{Y},$$

che può interrogare P e produce un'uscita $y \in \mathcal{Y}$ a partire da un input $x \in \mathcal{X}$. L'obiettivo è garantire che, anche per un avversario con accesso a P, l'uscita sia indistinguibile da un valore uniforme. Nel seguito, P rappresenterà la primitiva ideale di riferimento, che generalizza il comportamento dell'oracolo casuale G impiegato nella modellazione monolitica introdotta in precedenza.

L'avversario è modellato come una coppia di algoritmi $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, entrambi con accesso all'oracolo P. Il primo stadio $\mathcal{A}_1$ produce un input x e uno stato interno σ; il secondo stadio $\mathcal{A}_2$ riceve un valore y e deve stabilire se esso provenga dall'estrattore reale o da una distribuzione uniforme. Indichiamo con $\mathcal{L}_1$ e $\mathcal{L}_2$ le liste di query a P effettuate rispettivamente da $\mathcal{A}_1$ e $\mathcal{A}_2$.

A questo punto è possibile formalizzare il modello di sicurezza per l'estrazione seedless attraverso **tre definizioni fondamentali**: la prima descrive la struttura dell'avversario e la sua capacità di interrogare l'oracolo, la seconda specifica la condizione di legittimità della sorgente, e la terza stabilisce la proprietà di sicurezza richiesta all'estrattore.

Definizione 2.2. Un avversario $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ è detto un *q-attacker* se $|\mathcal{L}_1 \cup \mathcal{L}_2| \leq q$; è detto un *q-IT-attacker* se $|\mathcal{L}_1| \leq q$, ossia se la seconda fase può interrogare senza limiti l'oracolo P (o equivalentemente, conoscerne completamente la funzione).

Il gioco di sicurezza per la seedless extraction procede come segue:

1. Il *challenger* sceglie un bit casuale $b \in \{0, 1\}$ e una primitiva $P \leftarrow \mathcal{P}$;
2. L'avversario $\mathcal{A}_1$ interagisce con P e produce (σ, x);
3. Il challenger calcola $y_0 \leftarrow \mathsf{ext}^P(x)$ e un valore casuale indipendente $y_1 \leftarrow \mathcal{Y}$;
4. L'avversario $\mathcal{A}_2$ riceve (σ, y_b), continua a interrogare P e produce un bit di risposta b';
5. L'avversario vince se $b' = b$.

Il vantaggio dell'avversario è definito come:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{ext},P} = \left| \Pr[b' = b] - \tfrac{1}{2} \right|.$$

Questa quantità rappresenta la capacità di distinguere l'uscita dell'estrattore da un valore casuale. Un estrattore è sicuro se tale vantaggio è trascurabile, cioè se nessun avversario efficiente può fare meglio di un puro lancio di moneta.

Affinché il gioco sia significativo, si impone la seguente condizione di legittimità.

Definizione 2.3. Un avversario $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ è detto γ^*-*legittimo* se l'input X generato da $\mathcal{A}_1$ soddisfa

$$H_\infty(X \mid \Sigma, \mathcal{L}_1) \geq \gamma^*,$$

dove Σ è la trascrizione delle risposte dell'oracolo e $\mathcal{L}_1$ la lista delle query effettuate nella prima fase.

Questa condizione assicura che, anche conoscendo tutte le risposte ricevute dall'oracolo, la sorgente conservi almeno γ^* bit di imprevedibilità effettiva.

Definizione 2.4. Un algoritmo $\mathsf{ext}^P : \mathcal{X} \to \mathcal{Y}$ è detto un $(\gamma^*, q, \varepsilon)$-*seedless extractor* nel modello P se, per ogni avversario $\mathcal{A}$ che sia γ^*-legittimo e che effettui al più q interrogazioni a P, vale

$$\mathrm{Adv}_{\mathsf{ext}}^{\mathsf{ext},P}(\mathcal{A}) \leq \varepsilon.$$

In altri termini, anche un avversario che conosca lo stato interno, la trascrizione e tutte le interazioni con l'oracolo non può distinguere l'uscita dell'estrattore da un valore casuale, purché la sorgente iniziale conservi sufficiente entropia legittima.

In conclusione, la definizione di sicurezza appena introdotta mostra come la legittimità della sorgente – ossia la presenza di entropia residua sufficiente nonostante le interazioni con l'oracolo – sia la condizione essenziale per garantire un'estrazione seedless sicura. Adesso andiamo ad approfondire quelli che sono gli estrattori Seedless presi in considerazione.

Essi possono essere distinti in due categorie principali:

(a) **Estrattori monolitici**, basati su un'unica primitiva ideale, come un oracolo casuale. Essi rappresentano il caso teorico più semplice e costituiscono il punto di partenza per comprendere le garanzie e i limiti delle costruzioni seedless;

(b) **Estrattori online**, che accumulano progressivamente entropia in uno stato interno aggiornato a ogni passo. Questa architettura riflette più fedelmente il comportamento dei generatori pseudocasuali reali e consente di modellare la rigenerazione dell'entropia nel tempo.

Nel lavoro originale viene inoltre analizzato un modello alternativo di estrazione basato sulla modalità *Cipher Block Chaining* (CBC). Tale variante, pur concettualmente affine alle costruzioni online, risulta dimostrata insicura nel modello seedless, come descritto nella Sezione 3.4[7], a conferma del fatto che non tutte le strutture crittografiche tradizionalmente impiegate per la pseudocasualità sono adatte all'estrazione priva di seme.

Un'ulteriore categoria di interesse è costituita dalle **funzioni di derivazione di chiavi** (*Key-Derivation Functions, KDFs*), che applicano i principi dell'estrazione seedless alla derivazione sicura di chiavi da materiale entropico condiviso o parzialmente compromesso. In particolare, il paradigma *Extract-then-PRF*[7] rappresenta un esempio concreto di estrazione seedless computazionale, in cui l'estrazione corrisponde alla condensazione dell'entropia e la fase PRF alla produzione di uscite pseudocasuali.

2.3.2 Estrazione seedless con Oracolo Casuale Monolitico

Per comprendere in modo più concreto il funzionamento dell'estrazione seedless, è utile partire da un caso "ideale", in cui la primitiva crittografica stessa svolge il ruolo di estrattore. Questo scenario, detto **estrazione monolitica**, rappresenta la forma più semplice del modello seedless e costituisce il punto di partenza per le costruzioni più realistiche, come gli estrattori online e i PRNG privi di seme. Questa analisi si colloca nel *modello ideale*, in cui l'oracolo G è trattato come una funzione perfettamente casuale. L'obiettivo è fornire una base teorica per le proprietà di sicurezza che verranno poi adattate, nei paragrafi successivi, a primitive concrete nel modello standard.

Costruzione 1. Si consideri un oracolo casuale

$$G : \{0,1\}^n \to \{0,1\}^n,$$

scelto uniformemente tra tutte le funzioni possibili sul dominio di n bit. L'estrattore monolitico applica direttamente l'oracolo alla sorgente:

$$\mathsf{mono}^G(x) := G(x).$$

L'idea di fondo è che, se la sorgente X contiene un numero sufficiente di bit imprevedibili, l'uscita $G(X)$ sarà statisticamente indistinguibile da una stringa realmente casuale, anche per un avversario che possa interrogare lo stesso oracolo.

Sicurezza computazionale. Nel modello computazionale si assume che l'avversario disponga di risorse limitate e possa inviare al massimo q interrogazioni a G. Il vantaggio massimo dell'avversario, ossia la probabilità di distinguere $G(X)$ da un'uscita casuale, è limitato da:

$$\varepsilon \leq \frac{q}{2^{\gamma^*}},$$

dove γ^* rappresenta la min-entropia della sorgente. Il termine q indica la quantità di informazione che l'avversario può accumulare interrogando l'oracolo, mentre 2^{γ^*} quantifica lo spazio effettivo dei valori possibili di X. Il rapporto $q/2^{\gamma^*}$ esprime dunque la probabilità residua di successo di un distinguitore ottimale. In altre parole, ogni interrogazione riduce lievemente la sicurezza, ma l'effetto complessivo

[7] Il paradigma *Extract-then-PRF*, proposto da Krawczyk [18], prevede una fase di estrazione (*Extract*) che condensa l'entropia in un valore quasi uniforme, seguita da una funzione pseudocasuale (PRF) che espande tale valore in chiavi o flussi di output. L'HKDF, oggi standardizzato in numerosi protocolli, è la realizzazione più nota di questo schema.

rimane trascurabile se la sorgente mantiene un'adeguata quantità di entropia. Questo risultato formalizza la **robustezza computazionale** del modello monolitico: un oracolo ideale si comporta come un estrattore sicuro finché l'avversario opera entro limiti computazionali realistici.

Sicurezza informazionale. Se invece si considera un avversario non vincolato, che dopo aver scelto l'input può conoscere l'intera tabella dell'oracolo, si entra nel caso di **sicurezza informazionale**. In tale contesto, la probabilità di distinzione è limitata da:

$$\varepsilon \leq \tfrac{1}{2}\sqrt{\frac{2^{-(\gamma^*-n)}}{1-\rho}} + \rho, \qquad \text{dove } \rho = \frac{q}{2^n}.$$

Il parametro ρ rappresenta la frazione dello spazio degli input effettivamente esplorata tramite le q query dell'avversario. Quando q è molto minore di 2^n, la distribuzione di $G(X)$ rimane indistinguibile da quella di una variabile uniforme. Questa disuguaglianza garantisce la **sicurezza assoluta** del modello monolitico: anche un avversario con potenza di calcolo illimitata non ottiene un vantaggio significativo finché la sorgente conserva entropia pari almeno alla dimensione dell'input n.

Scelta dei parametri ed Interpretazione. Per rendere i risultati più concreti, si può considerare un oracolo con output di $n = 512$ bit e un avversario che possa effettuare fino a $q = 2^{80}$ interrogazioni. In questo scenario, una sorgente X che possieda circa $\gamma^* \approx 160$ bit di min-entropia garantisce un livello di sicurezza effettivo di circa **80 bit**. Questo esempio mostra come la sicurezza dipenda dal rapporto tra l'entropia della sorgente e la capacità computazionale dell'avversario: anche con un numero molto elevato di interrogazioni, un'adeguata quantità di entropia rende la distinzione statisticamente **irrilevante**.

Pur non essendo direttamente implementabile, esso dimostra che un oracolo perfettamente casuale può fungere da estrattore seedless purché la sorgente mantenga una sufficiente imprevedibilità. Questo risultato fornisce la base concettuale per il passaggio alle costruzioni online, nelle quali l'entropia viene accumulata e rigenerata in modo progressivo nel tempo.

2.3.3 Estrattori Online

Le costruzioni *online* estendono in modo naturale il modello monolitico, rendendolo applicabile a sorgenti reali e a primitive di hash pratiche. Nei sistemi concreti, infatti, le funzioni di hash o le cifrature ideali non possono elaborare input di lunghezza arbitraria in un solo passo: operano invece su blocchi di dimensione fissa (ad esempio $m = 256, 512, 1600$ bit), accumulando progressivamente l'entropia nello stato interno. Per questo motivo, gli estrattori — e più in generale i PRNG seedless — devono essere concepiti come processi iterativi, capaci di assorbire gradualmente casualità da molteplici sorgenti parziali.

Costruzione 2. Un **estrattore online** è definito da due algoritmi principali, che cooperano nella gestione dello stato interno e nella produzione dell'uscita:

$$\mathsf{ext} = (\mathsf{refresh}, \mathsf{finalize}).$$

Il primo, refresh, prende in ingresso lo stato corrente s e un blocco di input $x \in \{0,1\}^m$, producendo un nuovo stato aggiornato $s' \leftarrow \mathsf{refresh}^P(s,x)$; il secondo, finalize, prende lo stato s e genera l'uscita finale $y \in \{0,1\}^r$ secondo $y \leftarrow \mathsf{finalize}^P(s)$. In questa formulazione, il primo algoritmo realizza la funzione di aggiornamento, accumulando entropia nei vari passi di esecuzione, mentre il secondo estrae la stringa pseudocasuale finale. Un estrattore online che elabora blocchi di m bit e produce un'uscita di r bit è detto un (m,r)-estrattore online.

Sicurezza nel modello standard. La definizione di sicurezza per gli estrattori online segue lo stesso principio del modello monolitico, ma aggiunge un parametro per catturare la natura iterativa del processo. Oltre al numero q di query all'oracolo P, si considera anche il numero ℓ di chiamate alla funzione refresh effettuate dall'avversario, corrispondenti ai blocchi di input elaborati. Un avversario è quindi caratterizzato come un (q,ℓ)-attacker.

Formalmente, un algoritmo $\mathsf{ext}^P : \mathcal{X} \to \mathcal{Y}$ definito da due sottoprocedure $(\mathsf{refresh}, \mathsf{finalize})$ è detto un $(\gamma^*, q, \ell, \varepsilon)$-estrattore online se, per ogni avversario γ^*-legittimo che rispetta tali limiti di interazione con l'oracolo, vale:

$$\mathrm{Adv}_{\mathsf{ext}}^{\mathsf{ext},P}(\mathcal{A}) \leq \varepsilon.$$

Il parametro q misura la complessità dell'interazione con l'oracolo, mentre ℓ rappresenta il numero di blocchi di input elaborati, ossia il numero di chiamate alla funzione di aggiornamento. L'entropia complessiva della sorgente non è fornita in un singolo passo, ma si distribuisce su più iterazioni, e lo stato interno deve garantire che il livello di imprevedibilità residua non scenda mai sotto la soglia di sicurezza γ^*. In tal modo, ogni aggiornamento mantiene o incrementa l'entropia dello stato, assicurando che l'uscita finale rimanga indistinguibile da quella di un oracolo casuale.

Scelte pratiche e Interpretazione. Dal punto di vista pratico, gli estrattori online possono essere costruiti utilizzando gli stessi meccanismi strutturali impiegati nelle funzioni di hash e nei PRNG moderni. Le principali architetture di riferimento sono **Merkle–Damgård**, **Davies–Meyer** e **Sponge**, che si basano su funzioni di compressione o permutazioni iterate. Questi schemi, ampiamente adottati in algoritmi come SHA-2, SHA-3, BLAKE2 e Keccak, consentono di trattare input arbitrariamente lunghi accumulando l'entropia nello stato interno, esattamente come previsto dal modello di estrazione online.

Un caso di particolare rilievo è rappresentato da **HMAC**, che si fonda sul paradigma Merkle–Damgård ma con alcune modifiche che ne semplificano l'uso come estrattore seedless. Grazie alla sua ampia diffusione e alle solide garanzie di sicurezza, HMAC può essere reinterpretata come un estrattore online, con prestazioni pratiche e proprietà formali comparabili a quelle delle costruzioni teoriche basate su funzioni a spugna.

Per completezza, nel paper originale queste architetture sono anche presentate in forma grafica, come mostrato nella figura (v. Figura 2.1). Nel complesso, questi schemi mostrano come il principio teorico dell'accumulo progressivo di entropia possa essere realizzato in modo efficiente attraverso primitive già standardizzate.

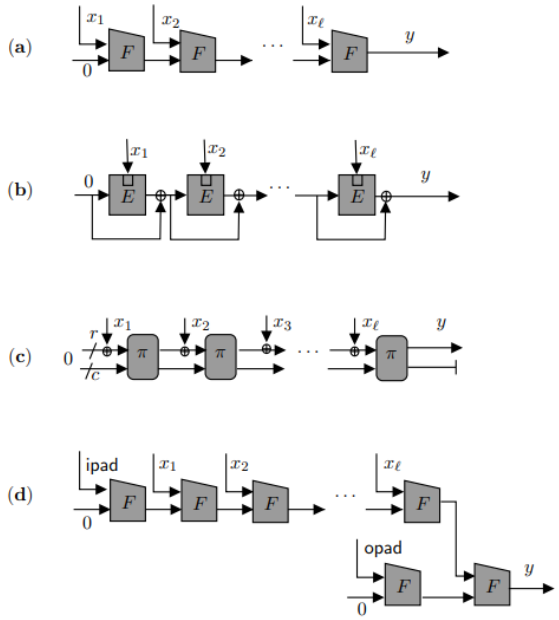


Figura 2.1. Quattro esempi di estrattori online computazionali: (a) Merkle–Damgård con funzione di compressione casuale, (b) Merkle–Damgård con schema Davies–Meyer, (c) Sponge, (d) HMAC. In ciascun caso, i blocchi di input $x_1, \ldots, x_\ell$ corrispondono alle chiamate successive della funzione refresh, mentre la fase finale finalize produce l'uscita y.

In questo modo si completa il quadro che stiamo cercando: mentre l'estrattore monolitico dimostra la possibilità teorica di ottenere casualità senza un seme indipendente, l'approccio online ne mostra la realizzabilità pratica, basata su aggiornamenti iterativi e primitive reali.

2.3.4 HKDF e paradigma Extract-then-PRF

La **HMAC-based Key-Derivation Function** (HKDF) [18] rappresenta la più nota istanziazione pratica del paradigma *Extract-then-Expand*, largamente impiegata per derivare chiavi segrete da sorgenti ad alta entropia in protocolli come TLS, IPsec e Noise.

Costruzione 3. Una *Key-Derivation Function* (KDF) è, in generale, un algoritmo che, a partire da un valore iniziale x (ad esempio un segreto condiviso o un output entropico parziale), genera una o più chiavi di uscita:

$$y \leftarrow \mathsf{KDF}(x, c, o),$$

dove c rappresenta un valore di contesto e o la lunghezza desiderata in bit. In questo senso, una KDF può essere vista come un estrattore che *raffina* l'entropia di una sorgente per produrre chiavi pseudocasuali e indipendenti, pronte per l'uso in protocolli crittografici.

Il paradigma *Extract-then-Expand* realizza tale processo in due fasi: una prima fase di *estrazione* (Extract) condensa l'entropia della sorgente in un valore quasi

uniforme, mentre una seconda fase di *espansione* (Expand) utilizza tale valore come chiave per una funzione pseudocasuale (PRF)[8] a lunghezza variabile. Nel caso di HKDF, entrambe le fasi sono implementate tramite la funzione HMAC[9], che agisce contemporaneamente come estrattore di entropia e come PRF standard. L'obiettivo di questa sezione è descrivere una versione *seedless* di HKDF, ottenuta sostituendo l'estrattore classico con un estrattore seedless all'interno dello stesso schema compositivo.

Sicurezza nel modello standard. La sicurezza delle KDF seedless è formalizzata come un esperimento tra un *challenger* e un avversario a due stadi $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, in modo analogo al modello di estrazione seedless. L'avversario riceve accesso oracolare alla funzione di derivazione e tenta di distinguere le sue uscite da valori realmente casuali. Il gioco procede come segue:

1. Il challenger sceglie un bit casuale $b \leftarrow \{0,1\}$;
2. $\mathcal{A}_1$ interagisce con l'oracolo e produce un input (σ, x);
3. $\mathcal{A}_2$ riceve σ e può formulare query di costruzione (c, o), che vengono risolte come $\mathsf{KDF}(x_1, \ldots, x_\ell, c, o)$ se $b = 0$, oppure con un valore uniforme $y_1 \leftarrow \{0,1\}^o$ se $b = 1$; $\mathcal{A}_2$ è limitato a una singola query per ogni c;
4. infine, $\mathcal{A}_2$ restituisce un bit b' e vince il gioco se e solo se $b' = b$.

Il vantaggio dell'avversario è definito come:

$$\mathrm{Adv}^{\mathsf{kdf}}_{\mathrm{KDF}}\mathcal{A} = \left| \Pr[b' = b] - \tfrac{1}{2} \right|.$$

Dire che una KDF è $(\mathcal{F}, T, q, \varepsilon)$-sicura significa che per ogni avversario appartenente alla classe di distribuzioni $\mathcal{F}$, che esegue al più q interrogazioni e opera entro tempo T, tale vantaggio è limitato da ε.

In questo modello, una **Seedless KDF** può essere costruita componendo un estrattore seedless e una funzione pseudocasuale a lunghezza variabile (VL-PRF):

$$\mathsf{KDF}(x, c, o) := \mathsf{PRF}(\mathsf{ext}(x), c, o).$$

Se ext è un $(\mathcal{F}, T, \varepsilon)$-estrattore seedless e PRF è una (T', q', ε')-VL-PRF, allora la funzione risultante è anch'essa una $(\mathcal{F}, \min(T, T'), q+q', \varepsilon+\varepsilon')$-*SL-KDF*. Ciò implica che la sicurezza di HKDF discende direttamente dalle due proprietà costitutive di HMAC: la pseudocasualità come PRF e la capacità di agire come estrattore seedless rispetto a sorgenti realistiche.

[8] Una *Pseudo-Random Function* è una funzione deterministica che, pur essendo calcolabile in modo efficiente, produce uscite indistinguibili da quelle di una funzione perfettamente casuale per ogni avversario computazionalmente limitato. Introdotta formalmente da Bellare e Rogaway [2], la PRF rappresenta una delle primitive fondamentali della crittografia simmetrica e costituisce la base teorica di MAC, KDF e PRNG moderni.

[9] La *Hash-based Message Authentication Code* (HMAC) è una costruzione standard per ottenere funzioni pseudocasuali a partire da una funzione di hash crittografica. Introdotta da Bellare, Canetti e Krawczyk [1], HMAC combina in modo sicuro una chiave segreta e un messaggio attraverso due applicazioni della funzione di hash, fornendo autenticazione e proprietà pseudocasuali ampiamente verificate sia nel modello standard che in quello ideale.

Scelte pratiche e Interpretazione. In termini più intuitivi, se HKDF non risultasse sicuro nel modello standard, ciò implicherebbe necessariamente che o HMAC non è una buona PRF, oppure non è un buon estrattore seedless — entrambe ipotesi incompatibili con i risultati noti in letteratura.

La sicurezza di HKDF rappresenta quindi una conferma empirica della validità del paradigma seedless e della sua applicabilità pratica. HKDF mostra come i principi teorici dell'estrazione e dell'accumulo di entropia possano tradursi in un meccanismo concreto di derivazione sicura di chiavi. Essa realizza un collegamento diretto tra gli estrattori seedless e i PRNG robusti: entrambi condividono l'obiettivo di mantenere la casualità anche dopo compromissioni parziali dello stato interno. In quest'ottica, HKDF costituisce la manifestazione pratica più diretta dei principi seedless, unendo la teoria dell'entropia computazionale con la robustezza e l'efficienza delle primitive hash standardizzate.

2.4 Definizione *formale* dei Pseudorandom Number Generators con Input

2.4.1 Sintassi formale dei PRNG con input

Nella Sezione 1.1 è stata introdotta la nozione generale di **pseudorandom number generator** (PRNG) come meccanismo in grado di produrre sequenze pseudocasuali a partire da una sorgente di entropia iniziale. Passiamo ora a una **definizione formale**, adattata al contesto del paradigma seedless, in cui il generatore è progettato per *accumulare entropia* nel tempo e mantenere la sicurezza anche dopo una compromissione parziale dello stato.

Definizione 2.5. (Sintassi dei PRNG con input)
Un *pseudorandom number generator with input* (PRNG) è una coppia di algoritmi

$$\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$$

che condividono uno stato interno $s \in \{0,1\}^n$ e hanno accesso a una primitiva ideale P.

- $\mathsf{refresh}$ prende in ingresso lo stato corrente s e un input entropico $x \in \{0,1\}^m$, producendo un nuovo stato aggiornato

$$s' \leftarrow \mathsf{refresh}^P(s, x);$$

- next prende lo stato s e restituisce una coppia formata da un nuovo stato e da un'uscita pseudocasuale

$$(s', y) \leftarrow \mathsf{next}^P(s), \qquad \textit{dove } y \in \{0,1\}^r.$$

Un PRNG che elabora blocchi di m bit e produce uscite di r bit è detto un (m, r)-*PRNG con input.*

Questa formulazione formalizza il comportamento iterativo dei generatori seedless. La funzione refresh implementa la fase di **assorbimento dell'entropia**, in cui lo stato interno viene aggiornato combinando l'informazione precedente con nuovi input potenzialmente parziali o correlati, mentre la funzione next realizza la **generazione pseudocasuale** e l'aggiornamento dello stato, producendo l'uscita che verrà esposta all'esterno del sistema.

Rispetto al modello tradizionale, questa struttura consente di rappresentare in modo naturale l'evoluzione di un generatore che opera nel tempo, integrando progressivamente entropia da fonti diverse. Il PRNG seedless può quindi essere visto come un'estensione diretta degli *estrattori online* introdotti nella Sezione 2.3.1, con l'aggiunta della fase di output, che permette di produrre bit pseudocasuali a ogni ciclo di aggiornamento. Questo schema dinamico, fondato sull'alternanza tra assorbimento e generazione, costituisce la base teorica per introdurre, nella sezione successiva, la nozione di **robustezza**.

2.4.2 Giochi di sicurezza e la robustezza dei SL-PRNG

Dopo aver definito la sintassi formale dei PRNG con input, è naturale chiedersi quale proprietà distingua un generatore realmente sicuro da uno che semplicemente elabora input entropici. Il cuore del modello è la nozione di **robustezza**, che cattura l'idea che un PRNG non debba limitarsi ad accumulare entropia, ma debba anche sopravvivere a compromissioni dello stato interno.

Un PRNG è **robusto** se è in grado di assorbire nuova entropia nel proprio stato, rigenerando casualità anche dopo una compromissione parziale o totale dello stato interno. In altri termini, la robustezza garantisce che, una volta accumulata entropia sufficiente, le uscite del generatore restino **indistinguibili da quelle di un PRNG ideale**, assicurando sia la *forward security* (sicurezza verso il futuro) che la *backward security* (sicurezza verso il passato) - già citate nella Sezione 1.2.

Gioco di robustezza. Prima di introdurre la nozione di robustezza, è utile distinguere tra le due principali forme di PRNG considerate nel paradigma seedless:

- Il **PRNG computazionale**, analizzato contro avversari con risorse limitate, la cui sicurezza si basa sulle proprietà computazionali della primitiva ideale P (come resistenza alle collisioni o alla preimmagine);
- Il **PRNG informazionale** (*Information-Theoretic PRNG, IT-PRNG*), che garantisce sicurezza anche contro avversari non limitati computazionalmente, fondando la propria robustezza unicamente sull'entropia residua della sorgente e sul comportamento ideale di P.

Queste due categorie riflettono direttamente le nozioni di sicurezza computazionale e informazionale introdotte per gli estrattori seedless (Sezione 2.3.1), ora estese al livello dei generatori completi. Nel seguito, entrambe le varianti saranno trattate in modo uniforme: le definizioni e i giochi di sicurezza introdotti per il caso computazionale si estendono naturalmente anche al caso informazionale, specificando di volta in volta le differenze rilevanti.

La robustezza viene formalizzata attraverso un esperimento (o gioco – *game*) tra un **challenger** e un **avversario adattivo** $\mathcal{A}$, che possono interagire mediante una famiglia di oracoli. Esso inizia scegliendo un bit casuale $b \in \{0,1\}$ e inizializzando lo stato del PRNG a $s \leftarrow 0^n$. L'avversario può poi interagire con gli oracoli elencati nella Tabella 2.1, che rappresentano le operazioni accessibili durante il gioco.

The PRNG Robustness Game		
init	**adv-refresh**(x)	**get-state**
$s \leftarrow 0^n$	$s \leftarrow \mathsf{refresh}^P(s,x)$	**return** s
$b \leftarrow \{0,1\}$		
next-ror	**get-next / get-next***	**set-state**(s^*)
$(s, y_0) \leftarrow \mathsf{next}^P(s)$	$(s,y) \leftarrow \mathsf{next}^P(s)$	$s \leftarrow s^*$
$y_1 \leftarrow \{0,1\}^r$	**return** y	
return y_b		

Tabella 2.1. Oracoli del gioco di robustezza per i PRNG seedless.

Gli oracoli modellano tutte le possibili interazioni tra l'avversario e il generatore:

- adv-refresh(x) aggiorna lo stato assorbendo l'input x;
- get-next e get-next* producono un'uscita pseudocasuale, con la differenza che get-next* può essere invocato anche prima che il PRNG abbia accumulato sufficiente entropia;
- next-ror restituisce l'uscita reale del PRNG se $b = 0$, o una stringa casuale se $b = 1$;
- get-state e set-state permettono rispettivamente di leggere o modificare lo stato interno del generatore.

Il vantaggio dell'avversario nel distinguere i due mondi è definito come:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{rob},P} = \left| \Pr[b' = b] - \tfrac{1}{2} \right|.$$

Un PRNG è tanto più sicuro quanto più questo valore è trascurabile.

Si analizzano ora le strategie dell'avversario. Distinguiamo due classi di attaccanti — **canonici** e **legittimi** — che catturano rispettivamente la struttura delle query e le ipotesi di entropia necessarie per garantire la significatività del modello.

Attaccanti canonici. Si definiscono innanzitutto gli *entropy drains*, ossia gli eventi che riducono l'entropia del PRNG, come:

- l'inizio del gioco;
- ogni chiamata a get-state o set-state;

- ogni invocazione di get-next*.

Un attaccante è detto *canonico* se non effettua query a get-next* né tenta di osservare lo stato immediatamente dopo un'operazione di adv-refresh. Ogni strategia non canonica può essere simulata da un attaccante equivalente, che ottiene lo stesso vantaggio; per questo motivo, da ora in poi, si assumono tutti gli avversari come canonici.

Attaccanti legittimi. Un attaccante è *legittimo* se formula query solo quando le sorgenti di input conservano una quantità sufficiente di entropia. La condizione è formalizzata considerando, per ogni passo i, le seguenti variabili:

- $\mathcal{L}_i$: la lista delle query all'oracolo P;
- Σ_i: lo stato interno dell'avversario;
- $\bar{X}_i$: gli input forniti dall'attaccante dopo l'ultimo *entropy drain*;
- S_i: lo stato del PRNG dopo l'ultimo *Most Recent Entropy Drain (MRED).*

Definizione 2.6 (Attaccanti legittimi)**.** Un attaccante $\mathcal{A}$ è detto γ^**-legittimo* se, per ogni istante i,

$$H_\infty(\bar{X}_i \mid \Sigma_i, \mathcal{L}_i, S_i) \geq \gamma^*.$$

Tale condizione assicura che la sorgente di input mantenga un livello minimo di imprevedibilità residua, anche conoscendo lo stato interno e la cronologia delle query.

Nel caso informazionale, la stessa condizione di entropia viene estesa includendo anche le chiamate di output (get-next e next-ror) tra gli eventi di *entropy drain*. Questo riflette lo scenario in cui l'avversario può osservare pienamente il comportamento del generatore e viene dunque modellato come un *IT-attaccante*.

PRNG robusti. Possiamo ora buttare giù la definizione formale di robustezza. Un attaccante è caratterizzato dai parametri:

- q: numero massimo di query all'oracolo P;
- ℓ: numero di chiamate a adv-refresh tra due *entropy drains* e una query di output;
- t: numero complessivo di chiamate a oracoli diversi da adv-refresh.

Definizione 2.7 (Robustezza dei PRNG)**.** Un generatore pseudocasuale con input $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ con accesso a una primitiva ideale P è detto $(\gamma^*, q, t, \ell, \varepsilon)$-*robusto* nel modello P se, per ogni attaccante γ^**-IT-legittimo* che rispetti tali limiti, vale:

$$\mathrm{Adv}^{\mathsf{rob},P}_{\mathrm{PRNG}}\mathcal{A} \leq \varepsilon.$$

In modo analogo agli estrattori online, un PRNG robusto può essere interpretato come una sequenza iterata di estrazioni seedless: ogni fase di refresh accumula entropia nello stato, mentre next produce un'uscita pseudocasuale indistinguibile da quella di un oracolo ideale. Questo legame stabilisce la continuità concettuale tra la teoria dell'estrazione e la costruzione dei generatori pseudocasuali privi di seme.

2.5 Costruzioni SL-PRNGs

In questa sezione si andranno a presentare tre costruzioni di PRNG **semplici, intuitive e pratiche**, che rappresentano versioni concretamente implementabili dei modelli introdotti finora:

- una costruzione basata sul paradigma **Merkle-Damgård (MD)** usando una funzione pubblica di compressione a lunghezza fissa;
- una costruzione basata sul paradigma **Merkle-Damgård con schema Davies-Meyer (MD-DM)** - come nello SHA-2 - e costruito a partire da qualsiasi cifrario a blocchi pubblico;
- una costruzione basata sul paradigma **Sponge** - come nello SHA-3 - il quale usa permutazioni pubbliche.

Per chiarezza, si mantengono costanti le seguenti convenzioni di notazione:

- n: dimensione dello stato interno del generatore;
- m: lunghezza di ciascun blocco di input entropico;
- r: numero di bit prodotti ad ogni chiamata di next;
- q: numero massimo di query all'oracolo ideale (F, E o π);
- ℓ: numero di invocazioni refresh tra due eventi di *entropy drain* e una query di output (non va confuso con il numero complessivo di iterazioni del generatore);
- t: numero complessivo di query diverse da refresh;
- γ^*: min–entropia residua garantita da ogni input valido.

Tali parametri verranno ripresi in tutte le costruzioni e nelle corrispondenti bound di sicurezza, sia nel caso computazionale sia in quello informazionale.

2.5.1 Costruzioni PRNG basate su Merkle–Damgård

Una prima realizzazione pratica di PRNG seedless può essere costruita a partire dal **paradigma Merkle–Damgård (MD)**, lo stesso principio su cui si basano molti algoritmi di hash, come SHA-1 e SHA-2. L'idea di fondo è utilizzare una funzione pubblica di compressione

$$F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n,$$

che combina lo stato interno e un nuovo blocco di input per generare un nuovo stato.

Questa funzione può essere vista come il cuore del processo di aggiornamento: ad ogni passo, lo stato accumula entropia dal nuovo blocco, condensandola in un valore di lunghezza fissata n. Su questa base si definiscono due varianti complementari: una che garantisce sicurezza *computazionale* e una che offre sicurezza *informazionale* (IT).

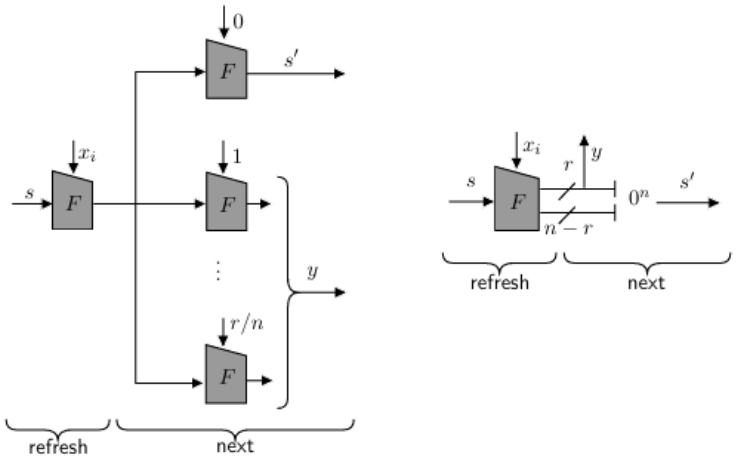


Figura 2.2. Procedure refresh (per l'elaborazione di un singolo blocco di input x_i) e next nelle costruzioni di PRNG basate sul paradigma Merkle–Damgård, con funzione di compressione F. A sinistra – costruzione computazionalmente sicura (4); a destra – costruzione informazionalmente sicura (5).

Costruzione 4 (Costruzione computazionale MD)**.** La prima costruzione, indicata come MD-PRNG, è definita dalle due procedure:

$$\mathsf{refresh}^F(s, x) = F(s, x), \qquad \mathsf{next}^F(s) = (F(s, 0),\, F(s, 1),\, \ldots,\, F(s, r/n)).$$

La procedura refresh aggiorna lo stato combinandolo con un nuovo input x, mentre next produce una sequenza di r/n blocchi, applicando iterativamente la funzione di compressione. In questo modo, il PRNG assorbe entropia dallo stato e genera output pseudocasuali di lunghezza arbitraria, mantenendo una struttura compatta e modulare identica a quella delle CHFs tradizionali.

La sicurezza di questa costruzione è analizzata nel cosiddetto **F-model**, in cui la funzione di compressione F è trattata come una funzione perfettamente casuale. In questo modello, F agisce come una **funzione casuale ideale**, ossia una mappa uniforme e indipendente su ogni possibile input. Ogni interrogazione restituisce un valore casuale coerente solo con le risposte precedenti, in modo analogo al *Random Oracle Model* di Bellare e Rogaway [2]. Questo approccio permette di analizzare la sicurezza del generatore in un contesto idealizzato, isolando le proprietà strutturali del paradigma Merkle–Damgård indipendentemente dai dettagli implementativi [10].

In questo contesto, la costruzione risulta un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-robusto, dove:

$$\varepsilon_{\text{rob}} \le 2t \cdot \left(\frac{\tilde{q}^2 + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma*}} \right), \qquad \textit{dove} \;\; \tilde{q} = q + \tfrac{r}{n} + 1.$$

Il termine ε_{rob} rappresenta il vantaggio massimo che un avversario può ottenere nel distinguere l'uscita del PRNG da quella di un generatore ideale. La relazione mostra che tale errore cresce proporzionalmente al numero di interazioni q e ℓ, ma decresce esponenzialmente con la dimensione dello stato n e con la min-entropia

γ^* della sorgente. Qui il termine $\tilde{q} = q + \frac{r}{n} + 1$ tiene conto anche delle iterazioni interne della funzione di compressione F, che contribuiscono al numero effettivo di chiamate all'oracolo ideale nel modello F.

Nel complesso, la formula evidenzia che la robustezza del PRNG aumenta al crescere della dimensione dello stato e dell'entropia disponibile, restando efficiente anche in presenza di numerosi cicli di aggiornamento.

Costruzione 5 (Costruzione informazionale MD)**.** Una variante informazionale dello stesso schema si ottiene modificando la procedura di generazione in modo che l'output della funzione next sia semplicemente la parte troncata dello stato, mentre il resto viene reinizializzato a zeri:

$$\mathsf{refresh}^F(s, x) = F(s, x), \qquad \mathsf{next}^F(s) = (0^n,\, s[1..r]).$$

In questo caso, il PRNG mantiene sicurezza perfetta anche se un avversario conosce l'intera tabella della funzione F, purché la sorgente di entropia fornisca un numero sufficiente di bit imprevedibili.

Questa seconda costruzione è un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob-it}})$-IT-robusto nel modello F, con:

$$\varepsilon_{\text{rob-it}} \leq \frac{t}{2}\sqrt{\frac{2^{r-\gamma^*}}{1-\rho} + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 \cdot \frac{\tilde{q}^2 2^r}{2^{2n}}} + t\rho \ ,$$

$$per\ \rho = \frac{\tilde{q}^2}{2^r}, \qquad dove\ \tilde{q} = q + t\ell$$

Il concetto è simile alla costruzione precedente. In questo modello però si assume che l'avversario possa conoscere interamente la funzione F, ma la sicurezza rimane garantita finché la sorgente fornisce almeno γ^* bit imprevedibili.

Il termine principale $2^{r-\gamma^*}$ rappresenta la perdita dovuta alla differenza tra la lunghezza dell'uscita e l'entropia effettivamente disponibile, mentre i termini proporzionali a 2^{-2n} e al parametro $\rho = \tilde{q}^2/2^r$ descrivono eventi di collisione e casi marginali che si verificano con probabilità trascurabile.

Inoltre, in questo caso $\tilde{q} = q + t\ell$ rappresenta il numero effettivo di interazioni con l'oracolo ideale F, includendo anche le chiamate interne generate dalle fasi di aggiornamento e di output. Sebbene la formula appaia complessa, essa esprime un comportamento asintotico stabile; la perdita di sicurezza cresce solo lentamente con ℓ, mantenendo un comportamento stabile anche dopo numerosi aggiornamenti dello stato.

Scelta dei parametri e interpretazione. Per una stima concreta, consideriamo l'uso di **SHA-512** come funzione di compressione, con $n = 512$ bit.

- Nel caso **computazionale**, ponendo $r = n$, $t = 1$, $q = 2^{80}$ e $\ell = \gamma^*$, si assume che da ogni blocco si ottenga almeno un bit di entropia. In queste condizioni, sono necessari circa $\gamma^* \simeq 163$ bit di entropia per ottenere una robustezza di circa 80 bit di sicurezza effettiva.

- Nel caso **informazionale**, con $n = 512$ e $r = 256$, e approssimando $1/(1-\rho) \leq 2$, si ottiene:

$$\varepsilon_{\text{rob-it}} \leq \frac{t}{2}\sqrt{2^{257-\gamma^*} + \frac{\ell \cdot d'(\ell)}{2^{256}} + t\frac{q^2}{2^{256}}}.$$

  Se poniamo $\ell = \gamma^*$ e, ad esempio, $q = 2^{80}$, è necessaria una perdita di entropia $\gamma^* - r = 162$ per ottenere circa 80 bit di sicurezza effettiva.

2.5.2 Costruzioni PRNG basate su Merkle–Damgård con schema Davies–Meyer

Una seconda realizzazione pratica del paradigma seedless si basa sulla **funzione di compressione Davies–Meyer (DM)**, utilizzata in molti hash come SHA-2. In questo caso, la funzione di compressione non è una mappa pubblica casuale, ma deriva da un *cifrario a blocchi pubblico* E, che opera su due input – una chiave e un messaggio – producendo un blocco di n bit. La compressione si ottiene calcolando:

$$E(b, a) \oplus a,$$

dove E è un cifrario pubblico scelto uniformemente tra tutti i cifrari possibili di chiave $b \in \{0,1\}^n$ e messaggio $a \in \{0,1\}^n$.

Nel corrispondente **E-model** (*ideal cipher model*), il cifrario E_k è trattato come una permutazione casuale e indipendente per ogni chiave k, e l'avversario può interrogare liberamente sia la funzione diretta $E_k(x)$ sia quella inversa $E_k^{-1}(y)$. Questa astrazione, introdotta formalmente da Dodis e Puniya [10], permette di analizzare la sicurezza delle costruzioni basate su cifrari a blocchi nello stesso modo in cui l'F-model tratta le funzioni di compressione.

Come mostrato in Figura 2.3, questo schema consente di costruire un PRNG seedless basato sul paradigma Merkle–Damgård–Davies–Meyer (MD-DM), che può essere definito in due varianti analoghe a quelle precedenti: una con sicurezza computazionale e una con sicurezza informazionale.

Costruzione 6 (Costruzione computazionale MD–DM)**.** La prima costruzione, indicata come DM-PRNG, è definita dalle seguenti procedure:

$$\mathsf{refresh}^E(s, x) = E(x, s) \oplus s, \qquad \mathsf{next}^E(s) = (E(0, s) \oplus s,\ E(1, s) \oplus s \,||\, \ldots \,||\, E(r/n, s) \oplus s).$$

La funzione refresh aggiorna lo stato combinandolo con l'input x tramite il cifrario E, mentre next genera l'output concatenando le applicazioni successive del cifrario su etichette crescenti. La struttura del PRNG resta compatta, ma introduce una *diffusione cifrante* sullo stato, che rafforza la resistenza a collisioni e preimmagini.

In tale modello, la costruzione risulta un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-robusto, con:

$$\varepsilon_{\text{rob}} \leq 4t \cdot \left(\frac{\tilde{q}^2 + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}}\right) \qquad \textit{dove } \tilde{q} = q + \tfrac{r}{n} + 1.$$

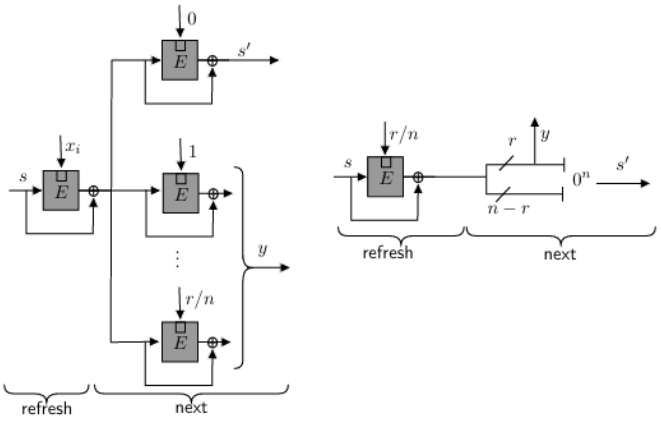


Figura 2.3. Procedure refresh (per l'elaborazione di un singolo blocco di input x_i) e next nelle costruzioni di PRNG basate sul paradigma Merkle–Damgård con compressione Davies–Meyer. A sinistra: costruzione computazionalmente sicura (Costruzione 6); a destra: costruzione informazionalmente sicura (Costruzione 7).

La formula mostra che, come nel caso Merkle–Damgård classico (Costruzione 4), l'errore di sicurezza cresce con il numero di query e il numero di fasi di aggiornamento, ma resta esponenzialmente piccolo rispetto alla dimensione del blocco n e all'entropia minima γ^* della sorgente. L'introduzione della struttura di cifratura riduce ulteriormente il rischio di correlazioni lineari nello stato interno, fornendo una robustezza computazionale più forte a parità di parametri.

Il termine $\tilde{q} = q + \frac{r}{n} + 1$ include anche le applicazioni interne del cifrario ideale E nella funzione di compressione, che incrementano il numero complessivo di query considerate nel modello E.

Costruzione 7 (Costruzione informazionale MD–DM)**.** La variante informazionale dello schema MD–DM mantiene la stessa procedura di aggiornamento, ma nella fase di generazione next produce solo i primi r bit dello stato, azzerando il resto:

$$\mathsf{refresh}^E(s, x) = E(x, s) \oplus s, \qquad \mathsf{next}^E(s) = (0^n,\, s[1..r]).$$

Anche in questo caso, il PRNG rimane sicuro contro avversari che conoscano completamente la struttura del cifrario, a condizione che la sorgente di input mantenga una quantità sufficiente di entropia.

Nel modello E, questa costruzione risulta un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob-it}})$-IT-robusto, dove:

$$\varepsilon_{\text{rob-it}} \le \frac{t}{2}\sqrt{\frac{2^{r-\gamma^*}}{1-\rho} + \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} + 64\ell^4\frac{2^r}{2^{2n-2}} + 16\ell^2\tilde{q}^2\frac{2^r}{2^{2n-2}}} + t\rho,$$

$$\textit{per } \rho = \frac{\tilde{q}^2}{2^r}. \qquad \textit{dove } \tilde{q} = q + t\ell$$

Il termine dominante $\frac{2^{r-\gamma^*}}{1-\rho}$ descrive la perdita di sicurezza dovuta alla differenza tra entropia effettiva e lunghezza dell'uscita, mentre i termini aggiuntivi modellano gli effetti cumulativi di collisioni e iterazioni multiple.

In questo caso, $\tilde{q} = q + t\ell$ tiene conto delle invocazioni interne del cifrario E dovute alle fasi di aggiornamento e generazione, riflettendo il numero totale di interazioni con l'oracolo ideale. La costruzione mantiene stabilità asintotica anche per grandi ℓ, assicurando che il vantaggio dell'avversario resti trascurabile finché la sorgente fornisce input a sufficiente entropia.

Scelta dei parametri e interpretazione. Per un esempio concreto, si può considerare l'uso di **SHA-512** come cifrario a blocchi di riferimento ($n = 512$).

- Nel caso **computazionale**, ponendo $r = n$, $t = 1$, $q = 2^{80}$ e $\gamma^* = 163$, si ottiene una robustezza di circa 80 bit di sicurezza effettiva.
- Nel caso **informazionale**, con $r = 256$, si approssima $1/(1-\rho) \leq 2$, ottenendo:

$$\varepsilon_{\text{rob-it}} \leq \frac{t}{2}\sqrt{2^{129-\gamma^*} + \frac{\ell \cdot d'(\ell)}{2^{127}}} + t\frac{q^2}{2^{128}} \ ,$$

  che fornisce un margine simile con $\gamma^* \approx 162$.

Nel complesso, il paradigma Merkle–Damgård–Davies–Meyer unisce la solidità del modello di compressione MD alla maggiore resistenza del cifrario a blocchi, estendendo la robustezza seedless sia in senso computazionale che informazionale.

2.5.3 Costruzioni PRNG basate su Sponge

Un'ulteriore istanziazione pratica del paradigma seedless è rappresentata dalle costruzioni basate su **Sponge**, lo stesso principio impiegato negli algoritmi SHA-3 e Keccak. Come nei casi precedenti, lo stato interno viene diviso in due parti: una porzione di *rate* (r) destinata all'assorbimento e all'emissione dei dati, e una porzione di *capacity* (c) dedicata alla sicurezza interna. Si ha quindi $n = r + c$.

Un PRNG Sponge si basa su una permutazione pubblica

$$\pi : \{0,1\}^n \to \{0,1\}^n,$$

che funge sia da funzione di aggiornamento che da generatore di uscita, seguendo il comportamento tipico delle funzioni "a spugna".

Costruzione 8 (Costruzione computazionale Sponge)**.** La costruzione $\mathsf{Spg} = (\mathsf{refresh}, \mathsf{next})$ utilizza una permutazione $\pi : \{0,1\}^n \to \{0,1\}^n$ per assorbire blocchi di r bit e produrre uscite di pari lunghezza:

$$\mathsf{refresh}^\pi(s, x) = \pi(s \oplus (x \,\|\, 0^c)), \qquad \mathsf{next}^\pi(s) = \big(\pi(s) \oplus (0^r \,\|\, s^{(c)}),\ s^{(r)}\big).$$

La procedura refresh combina lo stato s con l'input x nella parte di rate, mentre next esegue una singola chiamata alla permutazione per produrre sia l'output y che il nuovo stato s'. Questo approccio elimina la necessità di più chiamate parallele alla primitiva, rendendo la costruzione più efficiente rispetto allo schema Merkle–Damgård.

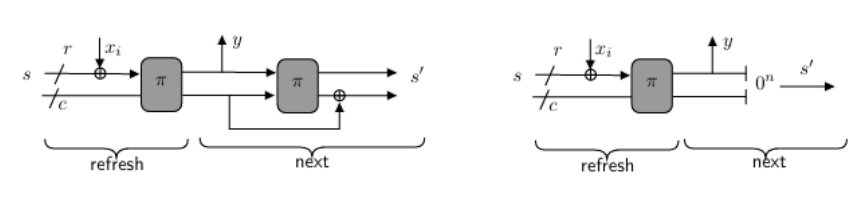


Figura 2.4. Procedure refresh (per l'elaborazione di un singolo blocco di input x_i) e next nelle costruzioni di PRNG basate sul paradigma Sponge, con permutazione pubblica π. A sinistra – costruzione computazionalmente sicura (8); a destra – costruzione informazionalmente sicura (9).

La sicurezza è analizzata nel cosiddetto π**-model**, un modello idealizzato in cui la permutazione π è considerata una funzione perfettamente casuale e biiettiva, scelta uniformemente tra tutte le permutazioni possibili su n bit. In questo contesto, l'avversario può interrogare la permutazione sia in avanti che all'indietro, simulando quindi un comportamento ideale. Il π-model è l'equivalente del *random oracle* e del *ideal cipher model* per le primitive basate su permutazioni, ed è comunemente utilizzato per analizzare la sicurezza di costruzioni come SHA-3 o Keccak, che si fondano sul paradigma Sponge [3, 6].

In tale contesto, la costruzione è un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-robusto, dove:

$$\varepsilon_{\text{rob}} \leq 4t \cdot \left(\frac{\tilde{q}^2 + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma*}} + \frac{\tilde{q}^2}{2^c} \right), \qquad \textit{dove } \tilde{q} = q + \tfrac{r}{n} + 1.$$

Il termine ε_{rob} rappresenta il vantaggio massimo dell'avversario nel distinguere l'uscita del PRNG da quella di un generatore ideale. I tre addendi principali esprimono rispettivamente la probabilità di collisione sullo stato, le fughe di informazione sul rate e gli effetti cumulativi sulla capacity. La robustezza cresce quindi all'aumentare dei parametri n e c, che controllano la dimensione e l'isolamento interno dello stato.

In questo caso, $\tilde{q} = q + \frac{r}{n} + 1$ tiene conto anche delle applicazioni interne della permutazione ideale π utilizzate durante i passaggi di assorbimento e di emissione, che contribuiscono al conteggio complessivo di query nel modello π.

Costruzione 9 (Costruzione informazionale Sponge)**.** La variante informazionale adotta la stessa funzione di aggiornamento, ma produce l'output troncando lo stato ai primi r bit e reinizializzandolo a zeri:

$$\mathsf{refresh}^{\pi}(s, x) = \pi(s \oplus x \,\|\, 0^c), \qquad \mathsf{next}^{\pi}(s) = (0^n,\, s[1..r]).$$

In questo modo, anche un avversario che conosca interamente la permutazione π non ottiene vantaggio, a condizione che la sorgente possieda sufficiente entropia.

Nel π-model, questa costruzione risulta un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob-it}})$-IT-robusto, con:

$$\varepsilon_{\text{rob-it}} \leq \frac{t}{2}\sqrt{\frac{2^{r-\gamma^*}}{1-\rho} + \frac{\ell \cdot (\ell + \tilde{q})}{2^{c-1}}} + t\rho, \qquad \textit{per } \rho = \frac{\tilde{q}^2}{2^c} \quad \textit{dove } \tilde{q} = q + t\ell.$$

Il termine $2^{r-\gamma^*}$ descrive la perdita di entropia effettiva, mentre i termini in 2^{-c} riflettono la protezione offerta dalla capacità. Una capacity più ampia riduce il rischio cumulativo di collisioni e mantiene la stabilità della sicurezza anche dopo numerosi cicli di aggiornamento.

In questa versione, $\tilde{q} = q + t\ell$ riflette il numero effettivo di interazioni con la permutazione ideale π, comprendendo sia le operazioni di aggiornamento che quelle di generazione dei bit di uscita.

Scelta dei parametri e interpretazione. Per una stima concreta, consideriamo parametri simili a quelli di **SHA-3**: $n = 1600$, $c = 1024$, quindi $r = 576$.

- Nel caso **computazionale**, con $t = 1$, $q = 2^{80}$ e $\ell = \gamma^*$, sono necessari circa $\gamma^* \approx 163$ bit per ottenere 80 bit di sicurezza effettiva.
- Nel caso **informazionale**, con gli stessi parametri, il sistema richiede una perdita di entropia di circa 160 bit per raggiungere lo stesso livello di sicurezza.

Le costruzioni Sponge offrono dunque un'ottima efficienza grazie all'uso di una singola permutazione per ogni ciclo, mantenendo al contempo un'elevata robustezza teorica, che cresce con la capacità interna c e con la min–entropia della sorgente.

2.6 Prove di sicurezza per le Costruzioni Computazionali

Questa sezione analizza le costruzioni di generatori seedless che offrono sicurezza computazionale, illustrandone il metodo di dimostrazione e i principali risultati. L'obiettivo è fornire una trattazione unificata delle prove di robustezza dei PRNG presentati in precedenza, mostrando come tali costruzioni risultino sicure anche in presenza di avversari computazionalmente potenti.

Il nucleo tecnico di tutte le dimostrazioni si basa sul cosiddetto **metodo dell'H-coefficient**, una tecnica che consente di valutare la distanza tra il comportamento reale di un generatore e quello di un oracolo perfettamente casuale. Tale metodo traduce il problema della robustezza in un'analisi probabilistica degli eventi che possono alterare l'indistinguibilità tra i due mondi, come collisioni o perdite di entropia.

Nota metodologica. Le dimostrazioni riportate in questa sezione riprendono fedelmente l'impostazione di Dodis et al. [7], ma sono presentate in forma semplificata e commentata. L'obiettivo è chiarire i principi logici e probabilistici che guidano ciascuna prova, omettendo i passaggi puramente combinatori o algebrici che non aggiungono valore esplicativo. I risultati principali e le formule chiave sono riportati inalterati rispetto al paper originale, mentre per i dettagli tecnici completi si rimanda alle dimostrazioni dei Lemmi e dei Teoremi citati nei riferimenti.

Nelle sezioni seguenti verranno presentati i risultati relativi a ciascuna costruzione computazionale. Dopo l'introduzione del metodo di dimostrazione, si mostreranno i teoremi di sicurezza per i tre paradigmi considerati: **Merkle–Damgård**, **Merkle–Damgård con Davies–Meyer** e **Sponge**, evidenziando per ciascuno i passaggi logici e le ipotesi alla base dei rispettivi bound di sicurezza.

2.6.1 La tecnica dell'H-coefficient

Nella sezione 2.3.0 sono stati introdotti i concetti di distanza statistica e vantaggio dell'avversario, che costituiscono la base per l'analisi formale della sicurezza dei modelli seedless. Presentiamo ora la **tecnica dell'H-coefficient**, uno strumento generale per limitare il vantaggio di un avversario nel distinguere tra due mondi sperimentali, detti *reale* e *ideale*. Questa tecnica costituisce il punto di partenza di tutte le prove di robustezza dei PRNG che verranno analizzate nelle sezioni successive.

Trascritti. Per analizzare formalmente l'interazione di un avversario adattivo $\mathcal{D}$ con un sistema, è utile rappresentare la sequenza di query e risposte generate durante l'esperimento sotto forma di un **trascritto** τ. Ogni esecuzione produce dunque un trascritto τ che descrive completamente le osservazioni dell'avversario.

Indichiamo con:

$$p_0(\tau) \quad \text{e} \quad p_1(\tau)$$

le probabilità che τ venga **generato** rispettivamente nell'esperimento *reale* e in quello *ideale*. Inoltre, sia $p_{\mathcal{D}}(\tau)$ la probabilità che $\mathcal{D}$ produca una determinata decisione dopo aver osservato τ. Possiamo allora esprimere la probabilità complessiva di ottenere τ come:

$$\Pr[T_0 = \tau] = p_{\mathcal{D}}(\tau) \cdot p_0(\tau), \qquad \Pr[T_1 = \tau] = p_{\mathcal{D}}(\tau) \cdot p_1(\tau),$$

dove T_0 e T_1 sono le variabili casuali che descrivono i trascritti nei due esperimenti. Il comportamento dell'avversario è quindi interamente descritto dalla distribuzione di tali trascritti e dalle probabilità associate a ciascuno di essi. D'ora in poi denotiamo $\mathcal{T}$ l'insieme di tutti i trascritti τ.

Limitare il vantaggio. Il vantaggio di un avversario adattivo $\mathcal{D}$ nel distinguere i due mondi è limitato dalla distanza statistica tra le distribuzioni dei trascritti:

$$\mathrm{SD}(T_0, T_1) = \sum_{\tau \in \mathcal{T}} \max\{0,\, \Pr[T_1 = \tau] - \Pr[T_0 = \tau]\}.$$

Espandendo le probabilità in funzione delle rispettive distribuzioni e del comportamento dell'avversario, si ottiene:

$$= \sum_{\tau \in \mathcal{T}} \max\{0,\, p_{\mathcal{D}}(\tau) \cdot p_1(\tau) - p_{\mathcal{D}}(\tau)\, p_0(\tau)\}.$$

Raccogliendo i termini comuni:

$$= \sum_{\tau \in \mathcal{T}} p_{\mathcal{D}}(\tau) \cdot p_1(\tau) \left(1 - \frac{p_0(\tau)}{p_1(\tau)}\right)$$

$$= \sum_{\tau \in \mathcal{T}} \Pr[T_1 = \tau] \left(1 - \frac{p_0(\tau)}{p_1(\tau)}\right).$$

Questa espressione mette in evidenza che la distanza statistica dipende unicamente dal rapporto $\frac{p_0(\tau)}{p_1(\tau)}$, che confronta la probabilità del medesimo trascritto nei due

mondi. Più questo rapporto si avvicina a 1, minore sarà la capacità dell'avversario di distinguere i due esperimenti.

Si consideri ora un sottoinsieme $\Gamma \subseteq \mathcal{T}$ di ***trascritti buoni***, per i quali è noto un limite inferiore

$$\frac{p_0(\tau)}{p_1(\tau)} \geq 1 - \varepsilon \qquad \text{per ogni } \tau \in \Gamma,$$

dove $\varepsilon \geq 0$. Sostituendo tale limite nella formula precedente, si ottiene:

$$\sum_{\tau \in \mathcal{T}} \Pr[T_1 = \tau] \left(1 - \frac{p_0(\tau)}{p_1(\tau)}\right) \leq \sum_{\tau \in \Gamma} \left(1 - \frac{p_0(\tau)}{p_1(\tau)}\right) + \sum_{\tau \in \mathcal{T} \setminus \Gamma} \Pr[T_1 = \tau],$$

e poiché per tutti i trascritti buoni $\left(1 - \frac{p_0(\tau)}{p_1(\tau)}\right) \leq \varepsilon$, segue immediatamente:

$$\sum_{\tau \in \mathcal{T}} \Pr[T_1 = \tau] \left(1 - \frac{p_0(\tau)}{p_1(\tau)}\right) \leq \varepsilon + \Pr[T_1 \in \mathcal{T} \setminus \Gamma].$$

In sintesi, il primo termine ε quantifica la differenza media tra mondo reale e ideale per i trascritti regolari, mentre il secondo termine misura la probabilità complessiva dei casi anomali in cui tale approssimazione non vale

I trascritti $\tau \in \mathcal{T} \setminus \Gamma$ sono comunemente detti ***trascritti cattivi***.[10] Applicando la tecnica dell'H-coefficient, l'analisi consiste quindi nel definire l'insieme dei trascritti buoni, limitare il fattore sopra riportato e dimostrare che i trascritti cattivi sono rari. Questo passaggio può essere condotto nel mondo ideale, in cui la derivazione risulta notevolmente più semplice rispetto al caso reale.

Definizione 2.8. Lemma dell'H-coefficient. Per due esperimenti descritti da $p_0(\cdot)$ e $p_1(\cdot)$, se esistono un insieme di trascritti buoni $\Gamma \subseteq \mathcal{T}$ e valori $\varepsilon, \delta \geq 0$ tali che:

1. *analisi del rapporto* (ratio analysis) $\frac{p_0(\tau)}{p_1(\tau)} \geq 1 - \varepsilon$ per ogni $\tau \in \Gamma$;
2. *analisi degli eventi rari* (bad events analysis) $\Pr[T_1 \notin \Gamma] \leq \delta$;

allora il vantaggio di distinzione di ogni avversario $\mathcal{D}$ è limitato da:

$$\mathrm{Adv}_{\mathcal{D}} \leq \varepsilon + \delta.$$

2.6.2 Estrattore Monolitico

Come introdotto nella Sezione 2.3.2, l'estrattore monolitico mono lavora con un oracolo casuale $G : \{0,1\}^m \to \{0,1\}^n$ e costituisce un esempio elementare di estrattore seedless. Questa sezione ne illustra la dimostrazione di sicurezza, che rappresenta una prima applicazione diretta della tecnica dell'H-coefficient introdotta nella Sezione 2.6.1.

[10] N.B. Nel contesto della tecnica dell'H-coefficient, i *good transcripts* rappresentano i casi in cui le probabilità dei due mondi (reale e ideale) rimangono quasi identiche, mentre i *bad transcripts* corrispondono agli eventi eccezionali in cui questa corrispondenza non può essere garantita. La loro influenza sul vantaggio dell'avversario è trascurabile, poiché si verificano con probabilità molto bassa.

L'obiettivo della prova è mostrare che l'estrattore mono, operante nel modello ideale G, è un $(\gamma^*, q, \varepsilon)$–estrattore con $\varepsilon \leq q/2^{\gamma^*}$. La dimostrazione segue in modo diretto lo schema generale dell'H-coefficient: si identificano i *trascritti buoni*, si calcola il rapporto di probabilità per tali trascritti (*ratio analysis*) e si limita la probabilità dei trascritti cattivi (*bad event analysis*).

Impostazione della prova. L'interazione tra un avversario adattivo $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ e il *challenger* del gioco d'estrazione può essere rappresentata mediante un trascritto:

$$\tau = (y^*, x, L),$$

dove:

- x è l'input fornito da $\mathcal{A}_1$;
- y^* è l'output del gioco (che può essere il valore reale dell'estrattore o una stringa casuale uniforme da $\{0,1\}^n$);
- $L = L_1 \cup L_2$ è la lista delle query eseguite da $\mathcal{A}_1$ e $\mathcal{A}_2$ verso l'oracolo F.

Un trascritto è detto *cattivo* (*bad transcript*) se una delle due parti $\mathcal{A}_1$ o $\mathcal{A}_2$ interroga l'oracolo su un input x già utilizzato nel processo di estrazione, ossia se la lista L contiene una coppia della forma $(x, *)$.

Per applicare il Lemma 2.8, occorre determinare:

1. il rapporto tra le probabilità dei trascritti buoni nei due mondi (reale e ideale) — analisi di tipo *ratio*;
2. la probabilità complessiva dei trascritti cattivi — analisi degli eventi rari (*bad events*).

Analisi del Rapporto. *Per ogni trascritto buono τ vale:*

$$\frac{p_0(\tau)}{p_1(\tau)} = 1.$$

Dimostrazione. Si fissi un trascritto buono τ e si consideri dapprima il caso ideale. Poiché in tale scenario y^* è estratto in modo uniforme da $\{0,1\}^n$, vale:

$$p_1(\tau) = p_L \cdot 2^{-n},$$

dove p_L rappresenta la probabilità che una funzione casuale sia consistente con le query contenute in L. Nel mondo reale, invece,

$$p_0(\tau) = p_L \cdot q_r,$$

dove q_r è la probabilità che $F_L(x) = y^*$, essendo F_L una funzione estratta uniformemente tra quelle compatibili con L. Poiché il trascritto è buono, F_L non è vincolata a x, e dunque $q_r = 2^{-n}$. Segue immediatamente che $p_0(\tau) = p_1(\tau)$ e quindi il rapporto è uguale a 1. □

Osservazione. Nel calcolo precedente, la quantità p_L non include la probabilità che x compaia nel trascritto. Come spiegato nella Sezione 2.6.1, le probabilità $p(\tau)$ sono sempre condizionate al fatto che l'avversario abbia prodotto le query indicate in τ, mentre l'eventuale apparizione di x è trattata separatamente come evento avverso (*bad event*).

Analisi degli Eventi Rari. *Sia $\mathcal{B}$ l'insieme dei trascritti cattivi.* Nel modello ideale vale:

$$\Pr[T_1 \in \mathcal{B}] \leq \frac{q}{2^{\gamma^*}}.$$

Dimostrazione. Dalla γ^*–legittimità dell'avversario $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ segue che:

$$H_\infty(X \mid \Sigma\mathcal{L}_1) \geq \gamma^*.$$

Nel mondo ideale, l'uscita dell'esperimento di estrazione è la stringa casuale Y^*, indipendente dall'input X generato da $\mathcal{A}_1$. È quindi possibile riscrivere l'ordine di campionamento come segue:

1. si estrae F uniformemente a caso;
2. $\mathcal{A}_1$ viene eseguito fino a produrre (σ, x) e la lista di query L_1;
3. si sceglie y^* uniformemente a caso;
4. si esegue $\mathcal{A}_2$ su (σ, y^*), ottenendo la lista di query L_2;
5. infine, si ricalcola X condizionato su $(\Sigma, L_1) = (\sigma, L_1)$.

Poiché la condizione include L_1, l'avversario $\mathcal{A}_1$ effettua le stesse query durante il primo e il secondo campionamento. La condizione di min–entropia rimane quindi valida anche per la lista completa $L = L_1 \cup L_2$:

$$H_\infty(X \mid \Sigma L) \geq \gamma^*.$$

Ne consegue che la probabilità che l'input X venga ripetuto all'interno di una query di L è al più $q \cdot 2^{-\gamma^*}$, da cui la tesi.

Come mostrato in Figura 2.5, questa analisi collega le diverse nozioni intermedie di sicurezza (*Extraction Security*, *Next Security*, *Maintaining Security*) alla nozione complessiva di **robustezza**. Le dimostrazioni che seguono riprendono esattamente la stessa struttura per le costruzioni PRNG seedless.

2.6.3 Nozioni intermedie di Sicurezza dei PRNG

Per analizzare la sicurezza dei PRNG nel paradigma seedless, è utile scomporre il concetto generale di **robustezza** in proprietà più elementari, ciascuna delle quali cattura un diverso aspetto del comportamento del generatore. Seguendo la tradizione della letteratura sui PRNG, si considerano due proprietà fondamentali: *recovering security* e *preserving security.*

La *Recovering security* garantisce che, dopo un evento di perdita di entropia (entropy drain), l'assorbimento di una quantità sufficiente di entropia nello stato

The PRNG Recovering Game	
init	**chall**$(s_0, x_1, \ldots, x_\ell)$
$b \leftarrow \{0,1\}$	**for** $i = 1, \ldots, \ell$ **do**
	$s_i \leftarrow \mathsf{refresh}(s_{i-1}, x_i)$
	if $b = 0$ **then**
	return $\mathsf{next}(s_\ell)$
	else
	$(s, y) \leftarrow \{0,1\}^{n+r}$
	return (s, y)

Tabella 2.2. Oracoli del gioco di recupero (*Recovering Game*) per i PRNG seedless.

permetta al PRNG di *recuperare la casualità*: in particolare, l'uscita della funzione next diventa indistinguibile da una stringa casuale uniforme in $\{0,1\}^{n+r}$. La *Preserving security*, invece, assicura che, anche in presenza di input scelti in modo avversario, uno stato ad alta entropia non venga degradato e che l'uscita successiva del generatore resti indistinguibile da un valore uniforme.

Queste due proprietà costituiscono la base per la dimostrazione della robustezza dei PRNG seedless, poiché insieme ne garantiscono la sicurezza composizionale. La loro combinazione sarà formalizzata nel Teorema 2.6.3 più avanti. Per dimostrare recovering e preserving security, si introducono inoltre tre proprietà più fini, che consentono di isolare le diverse componenti probabilistiche del modello:

- *Extraction security* — lo stato del PRNG è indistinguibile da una distribuzione uniforme dopo che è stata assorbita una quantità sufficiente di entropia;
- *Maintaining security* — uno stato già casuale rimane indistinguibile da una distribuzione uniforme anche dopo input arbitrari;
- *Next security* — l'uscita della funzione next, applicata a uno stato ad alta entropia, è indistinguibile da un valore casuale uniforme.

In sintesi:

1. le proprietà di *extraction* e *next* security **implicano** la *recovering security*;
2. le proprietà di *maintaining* e *next* security **implicano** la *preserving security*.

La Figura 2.5 illustra graficamente le relazioni di implicazione tra le varie nozioni di sicurezza.

Recovering security. L'intuizione alla base della recovering security è che, dopo un evento di perdita di entropia, una volta che un'adeguata quantità di entropia è stata riassorbita nello stato del PRNG, l'uscita di next torna ad essere indistinguibile da una stringa uniforme in $\{0,1\}^{n+r}$.

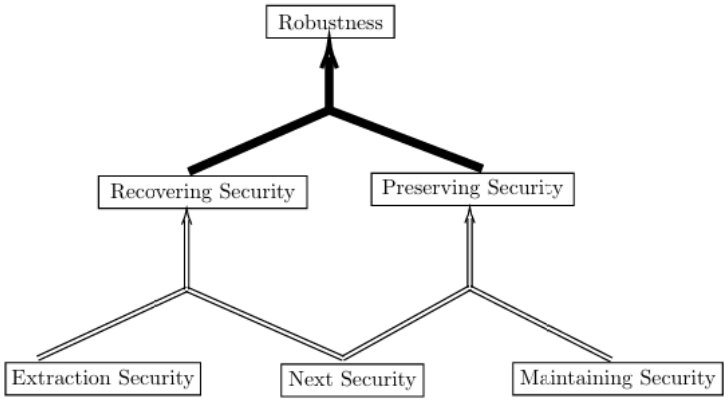


Figura 2.5. Relazioni di implicazione tra le diverse nozioni intermedie di sicurezza. Le frecce piene rappresentano una dimostrazione generica di implicazione, mentre quelle vuote indicano una prova specifica legata a una particolare costruzione.

Il gioco corrispondente è riportato nella Tabella 2.2. In tale esperimento, l'avversario specifica uno stato iniziale s_0 e una sequenza di input $x_1, \ldots, x_\ell$; gli input vengono assorbiti uno alla volta, e la funzione next viene infine chiamata sullo stato risultante. Il gioco restituisce l'uscita reale se $b = 0$ e una stringa casuale uniforme se $b = 1$. Il vantaggio di un avversario $\mathcal{A}$ in questo gioco è denotato con $\mathrm{Adv}^{\mathrm{rec},P}_{\mathrm{PRNG}}(\mathcal{A})$.

Preserving security. La preserving security richiede che, anche in presenza di input scelti adversarialmente, uno stato ad alta entropia non venga compromesso e che l'uscita successiva del PRNG rimanga indistinguibile da una stringa uniforme. Il gioco corrispondente è illustrato nella Tabella 2.3. In questo esperimento, l'avversario fornisce una sequenza di input che viene assorbita nello stato del PRNG; il gioco restituisce l'output reale di next se $b = 0$, oppure una stringa casuale uniforme se $b = 1$. Il vantaggio dell'avversario $\mathcal{A}$ è denotato da $\mathrm{Adv}^{\mathrm{pre},P}_{\mathrm{PRNG}}(\mathcal{A})$.

The PRNG Preserving Game

init	**chall**$(x_1, \ldots, x_\ell)$
$s_0 \leftarrow \{0,1\}^n \quad b \leftarrow \{0,1\}$	**for** $i = 1, \ldots, \ell$ **do**
	$s_i \leftarrow \mathsf{refresh}(s_{i-1}, x_i)$
	if $b = 0$ **then**
	return $\mathsf{next}(s_\ell)$
	else
	$(s, y) \leftarrow \{0,1\}^{n+r}$
	return (s, y)

Tabella 2.3. Oracoli del gioco di preservazione (*Preserving Game*) per i PRNG seedless.

Prima di passare alle definizioni formali, si specificano i vincoli di legittimità e i parametri che caratterizzano gli avversari ammessi nei due giochi.

Analogamente al gioco di robustezza (v. 2.4.2), l'avversario deve rispettare una condizione di legittimità γ^*, secondo la quale:

$$H_\infty(X_1, \ldots, X_\ell \mid \Sigma LS_0) \geq \gamma^*,$$

dove:

- Σ rappresenta lo stato interno dell'avversario immediatamente prima della chiamata all'oracolo **chall**;
- $\mathcal{L}$ è la lista delle query e risposte effettuate all'oracolo P prima della chiamata;
- S_0 è lo stato iniziale fornito dall'avversario.

Nel gioco di recovering, q denota il numero massimo di query che l'avversario può effettuare all'oracolo P, mentre ℓ rappresenta il numero massimo di blocchi di input assorbiti durante l'esperimento. Un avversario che rispetta questi limiti è detto (q, ℓ)–**attaccante**.

Definizione 2.9 (Sicurezza di recupero)**.** Un PRNG $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ è detto $(\gamma^*, q, \ell, \varepsilon)$–*recovering* nel modello P se, per ogni avversario (q, ℓ)–legittimo, vale:

$$\mathrm{Adv}^{\mathrm{rec},P}_{\mathrm{PRNG}}(\mathcal{A}) \leq \varepsilon.$$

Definizione 2.10 (Sicurezza di preservazione)**.** Un PRNG $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ è detto (q, ℓ, ε)–*preserving* nel modello P se, per ogni avversario (q, ℓ)–legittimo, vale:

$$\mathrm{Adv}^{\mathrm{pre},P}_{\mathrm{PRNG}}(\mathcal{A}) \leq \varepsilon.$$

Teorema di composizione. Sia $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ una costruzione tale che $\mathsf{refresh}$ effettui α chiamate all'oracolo P e next effettui β chiamate. Se il PRNG è:

- $(\gamma^*, q, \ell, \varepsilon_{\mathrm{rec}})$–recovering, e
- $(q, \ell, \varepsilon_{\mathrm{pre}})$–preserving,

allora esso è anche $(\gamma^*, q, t, \ell, \varepsilon_{\mathrm{rob}})$–robusto nel modello P, con:

$$\varepsilon_{\mathrm{rob}} \leq t \cdot (\varepsilon_{\mathrm{rec}} + \varepsilon_{\mathrm{pre}}).$$

Extraction, Maintaining e Next Security.

Le tre proprietà intermedie di *extraction*, *maintaining* e *next security* forniscono una base modulare per dimostrare le proprietà di *recovering* e *preserving security* nelle diverse costruzioni di PRNG.

Extraction security. La proprietà di extraction richiede che, dopo l'assorbimento di blocchi di input con elevata entropia congiunta, lo stato del PRNG risulti indistinguibile da una distribuzione uniforme. Il gioco corrispondente è una variante del *recovering game* (cfr. Figura 2.2), in cui la funzione next non viene applicata a s_ℓ: invece, il gioco restituisce s_ℓ oppure una stringa campionata uniformemente, a seconda del bit segreto $b \in \{0,1\}$. La legittimità di un avversario $\mathcal{A}$, insieme ai parametri q e ℓ, è definita come nel caso del recovering game. Il vantaggio di $\mathcal{A}$ contro la proprietà di extraction security nel modello P è indicato con $\mathrm{Adv}^{\mathrm{ext},P}_{\mathrm{PRNG}}(\mathcal{A})$.

Maintaining security. La proprietà di maintaining security è una variante del *preserving game* (cfr. Figura 2.3) in cui, anche in questo caso, la funzione next non è applicata a s_ℓ: il gioco restituisce quindi s_ℓ oppure un valore uniforme, a seconda del bit b. I parametri q e ℓ sono definiti in modo analogo a quelli della preserving security. Il vantaggio dell'avversario $\mathcal{A}$ nel modello P è denotato da $\mathrm{Adv}^{\mathrm{mnt},P}_{\mathrm{PRNG}}(\mathcal{A})$.

Next security. Infine, la proprietà di next security stabilisce che l'uscita della funzione next, applicata a uno stato uniformemente distribuito, sia indistinguibile da una stringa casuale uniforme. In questo gioco, un avversario $\mathcal{A}$ effettua al più q query alla primitiva ideale P e tenta di distinguere $\mathsf{next}^P(S)$ da un valore casuale U_{n+r}, per uno stato S scelto uniformemente in $\{0,1\}^n$. Il vantaggio dell'avversario è indicato con $\mathrm{Adv}^{\mathrm{next},P}_{\mathrm{PRNG}}(\mathcal{A})$.

2.6.4 PRNGs basati su Merkle–Damgård

In questa sezione si dimostra la **robustezza** della costruzione di PRNG basata sul paradigma **Merkle–Damgård**, già presentata nella Sezione 2.5.1. Si ricordi che la costruzione MD-PRNG $=$ (refresh, next) utilizza una funzione pubblica di compressione $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ e che la sua definizione formale è riportata in Costruzione 4.

L'obiettivo di questa sezione è mostrare che tale costruzione risulta un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\mathrm{rob}})$-robusto nel modello F, con margine di sicurezza pari a:

$$\varepsilon_{\mathrm{rob}} \le 2t \cdot \left(\frac{\tilde{q}^{\,2} + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}} \right), \qquad \text{dove} \quad \tilde{q} = q + \frac{r}{n} + 1.$$

Il risultato segue come caso particolare del Teorema 2.6.3, dimostrando separatamente le proprietà di **extraction**, **maintaining**, **next**, **recovering** e **preserving** security. In conclusione, la combinazione delle cinque proprietà intermedie dimostrate nelle sezioni seguenti consente di limitare il vantaggio dell'avversario, confermando la robustezza della costruzione MD-PRNG nel modello ideale F, in linea con i risultati di Dodis et al. [7].

Extraction Security – MD

La prima proprietà di sicurezza da verificare per la costruzione Merkle–Damgård è la **extraction security**, che garantisce che, dopo l'assorbimento di blocchi di input con sufficiente entropia, lo stato del PRNG risulti indistinguibile da una distribuzione

uniforme. In termini intuitivi, il generatore riesce a "estrarre" casualità effettiva dai dati assorbiti, rendendo il proprio stato interno imprevedibile anche per un avversario che osservi e controlli le query all'oracolo F.

Secondo il **Lemma 15** del paper (cfr. Sezione 6.4.1 [7]), il vantaggio di un avversario $\mathcal{A}$ (q, ℓ)–legittimo nel distinguere i due mondi (reale e ideale) è limitato da:

$$\mathrm{Adv}_{\mathrm{MD}}^{\mathrm{ext},F}(\mathcal{A}) \leq \varepsilon_{\mathrm{MD}}^{\mathrm{ext}}(\gamma^*, q, \ell) := \frac{q^2 + q\ell + \ell^2}{2^n} + \frac{2q}{2^{\gamma^*}}$$

dove il primo termine rappresenta la probabilità di collisioni o incoerenze nelle query a F, mentre il secondo quantifica la perdita di sicurezza dovuta a una sorgente di input con entropia limitata.

Setup e notazioni. Continuando con il lemma sopracitato, si considera un trascritto

$$\tau = (s^*, s_0, x_1, \ldots, x_\ell, L),$$

che descrive l'interazione tra l'avversario $\mathcal{A}$ e l'oracolo. Qui s_0 è lo stato iniziale scelto dall'avversario, s^* è l'uscita finale dell'esperimento, L è la lista delle query a F, e $x_1, \ldots, x_\ell$ sono i blocchi assorbiti.

Sia ℓ' il più grande indice tale che $((y_{i-1}, x_i), y_i) \in L$ per alcuni $y_0, \ldots, y_{\ell'}$ con $y_0 = s_0$. Il trascritto è *cattivo* se $\ell' = \ell$, ossia se l'avversario ha interrogato F su tutte le coppie necessarie a ricostruire s^*; in caso contrario, è *buono*. L'obiettivo è ora valutare la distanza statistica tra i due mondi attraverso la tecnica dell'H–coefficient.

Analisi del Rapporto. Ricordiamo che per ogni trascritto buono τ, vale

$$\frac{p_0(\tau)}{p_1(\tau)} \geq 1 - \frac{q\ell + \ell^2}{2^n}.$$

Dimostrazione. Si fissi un trascritto buono τ e si considerino le due probabilità nei mondi reale e ideale. Nel mondo ideale, l'uscita s^* è campionata uniformemente, per cui

$$p_1(\tau) = p_L \cdot 2^{-n},$$

dove p_L è la probabilità che una funzione casuale uniforme risulti coerente con l'insieme di query L. Nel mondo reale, invece,

$$p_0(\tau) = p_L \cdot q_\tau,$$

dove

$$q_\tau = \Pr[\mathsf{MD}_{s_0}^{F_L}(x_1, \ldots, x_\ell) = s^*].$$

Il termine q_τ rappresenta la probabilità che la catena Merkle–Damgård calcolata a partire da s_0 e condizionata alle query in L produca proprio s^*.

Poiché τ è buono, esiste un indice $\ell' < \ell$ tale che gli stati intermedi $y_{\ell'+1}, \ldots, y_\ell$ non sono determinati dalle query osservate. Scriviamo allora

$$Y_\ell := \mathsf{MD}_{y_{\ell'}}^{F_L}(x_{\ell'+1}, \ldots, x_\ell) = s^*.$$

Definiamo per ciascun passo intermedio l'evento di freschezza

$$\mathsf{FRESH}_i := \text{“ nessuna query a } F(Y_i, *) \text{ è presente in } L \text{ e } Y_i \neq Y_j \text{ per } j < i \text{ ”}$$

e poniamo

$$\mathsf{FRESH} := \bigcap_{i=\ell'+1}^{\ell-1} \mathsf{FRESH}_i.$$

Condizionato all'evento FRESH, il valore Y_ℓ è campionato uniformemente, quindi

$$\Pr[Y_\ell = s^* \mid \mathsf{FRESH}] = 2^{-n}.$$

Rimane da valutare la probabilità che l'evento di freschezza si verifichi.

Nel paper si effettua un conteggio combinatorio dettagliato sulle possibili coppie "non fresche" generate dalle query a F. Il ragionamento mostra che, mentre vengono generati gli stati intermedi Y_i, esistono al più $q + \ell$ valori "occupati" (già usati come input di F), e quindi:

$$\Pr\Big[\bigcap_{k=\ell'+1}^{i-1} \mathsf{FRESH}_k\Big] \leq \frac{q+\ell}{2^n}.$$

Poiché questo vale per ogni passo, si ottiene il bound aggregato:

$$\Pr[\mathsf{FRESH}] \geq 1 - \frac{q\ell + \ell^2}{2^n}.$$

(*Nella trattazione originale viene introdotta anche la notazione $\overline{A}$ per l'evento complementare e mostrato come combinare le probabilità tramite un'argomentazione di tipo unione; qui tali passaggi tecnici sono omessi per brevità*).

Combinando i risultati precedenti si ottiene:

$$\begin{aligned} q_\tau = \Pr[Y_\ell = s^*] &= \Pr[\mathsf{FRESH}] \cdot \Pr[Y_\ell = s^* \mid \mathsf{FRESH}] \\ &\geq \Big(1 - \frac{q\ell + \ell^2}{2^n}\Big) \cdot 2^{-n}. \end{aligned}$$

Sostituendo le espressioni di $p_0(\tau)$ e $p_1(\tau)$, si ottiene infine:

$$\frac{p_0(\tau)}{p_1(\tau)} = \frac{q_\tau}{2^{-n}} \geq 1 - \frac{q\ell + \ell^2}{2^n}.$$

Questo risultato mostra che, per tutti i trascritti buoni, le due distribuzioni (reale e ideale) differiscono solo di una quantità trascurabile, proporzionale alla probabilità di collisione tra le query all'oracolo F. I dettagli combinatori completi sono riportati nel **Lemma 16** nella Sezione 6.4.1 [7].

Analisi degli Eventi Rari. Si considera ora la probabilità che un trascritto τ sia *cattivo*. Nel mondo ideale ($b = 1$), l'uscita del gioco di estrazione è un valore s^* campionato uniformemente, indipendente dallo stato iniziale s_0 e dagli input $x_1, \ldots, x_\ell$.

Dimostrazione. L'ordine di campionamento dell'esperimento può quindi essere descritto come segue:

1. Campiona F in modo uniforme.
2. Esegui $\mathcal{A}$ fino a quando produce $(\tilde{s}_0, \tilde{x}_1, \ldots, \tilde{x}_{\tilde{\ell}})$, generando anche lo stato σ e la lista di query L_0.
3. Scegli s^* uniformemente a caso.
4. Continua l'esecuzione di $\mathcal{A}$ su σ e s^*, consentendole di effettuare ulteriori query L_1.
5. Riesegui il campionamento degli input $(X_1, \ldots, X_\ell)$ condizionato su $(\Sigma, L_0, S_0) = (\sigma, L_0, s_0)$.

Sia $\mathcal{L}$ la variabile casuale che rappresenta l'insieme delle query totali fatte da $\mathcal{A}$. Poiché la condizione di legittimità γ^* si mantiene anche dopo il resampling, vale:

$$H_\infty(X_1, \ldots, X_\ell \mid \Sigma \mathcal{L} S_0) \geq \gamma^*.$$

Definiamo $\mathcal{E}$ come l'evento che esista una collisione in $\mathcal{L}$, ossia che due o più query abbiano lo stesso argomento. Si ha dunque:

$$\Pr[T_1 \in \mathcal{B}] \leq \Pr[T_1 \in \mathcal{B} \mid \mathcal{E}] + \Pr[\bar{\mathcal{E}}] \leq \Pr[T_1 \in \mathcal{B} \mid \mathcal{E}] + \frac{q^2}{2^n}.$$

Il primo termine rappresenta la probabilità di ottenere un trascritto cattivo in assenza di collisioni, mentre il secondo è la probabilità che una collisione si verifichi tra le q query all'oracolo F.

Per limitare il primo termine, si considera una tupla $z = (\sigma, L_0, s_0) \in \mathcal{E}$ e si denota con Z la variabile casuale corrispondente. Condizionato su $Z = z$, il gioco è cattivo se, per una delle catene possibili, i blocchi campionati coincidono con una sequenza determinata $(v_1, \ldots, v_\ell)$. Per la legittimità di $\mathcal{A}$,

$$\Pr[T_1 \in \mathcal{B} \mid Z = z] \leq q \cdot p_z, \qquad \textit{dove} \;\; p_z := \mathrm{Pred}(\overline{X} \mid Z = z),$$

dove p_z è la probabilità di "predire" la sequenza $\overline{X}$ di input non casuali dato $Z = z$.

In media, rispetto alla distribuzione di Z, vale:

$$\begin{aligned}\Pr[T_1 \in \mathcal{B} \mid \mathcal{E}] &= \sum_{z \in \mathcal{E}} \Pr[Z = z \mid \mathcal{E}] \cdot \Pr[T_1 \in \mathcal{B} \mid Z = z] \\ &\leq \sum_{z \in \mathcal{E}} \Pr[Z = z \mid \mathcal{E}] \cdot q \cdot p_z = q \cdot \mathrm{Pred}(\overline{X} \mid Z\mathcal{E}).\end{aligned}$$

Usando il vincolo di entropia γ^* e il fatto che $q^2/2^n \leq 1/2$, si ottiene:

$$\Pr[T_1 \in \mathcal{B} \mid \mathcal{E}] \leq \frac{q}{(1 - q^2/2^n)\, 2^{\gamma^*}} \leq \frac{2q}{2^{\gamma^*}}.$$

Combinando i due contributi si ottiene il bound complessivo:

$$\Pr[T_1 \in \mathcal{B}] \leq \frac{2q}{2^{\gamma^*}} + \frac{q^2}{2^n}.$$

Il primo termine è dovuto alla perdita di entropia residua e decresce esponenzialmente con γ^*, mentre il secondo corrisponde alla probabilità di collisione fra le query all'oracolo F. Per una derivazione completa delle disuguaglianze intermedie, si veda il **Lemma 17** in Sezione 6.4.1 [7].

Maintaining Security – MD

La **maintaining security** garantisce che, partendo da uno stato già uniforme, l'assorbimento di nuovi input non riduca l'imprevedibilità complessiva del PRNG. In altri termini, anche dopo interazioni avversarie, lo stato interno rimane indistinguibile da una distribuzione casuale.

Per ogni avversario (q, ℓ)–legittimo vale:

$$\mathrm{Adv}_{\mathrm{MD}}^{\mathrm{mnt},F}(\mathcal{A}) \leq \varepsilon_{\mathrm{MD}}^{\mathrm{mnt}}(q,\ell) := \frac{q\ell + \ell^2}{2^n} + \frac{q}{2^n}.$$

Dimostrazione. Anche questa prova segue la tecnica dell'H–coefficient. Si considera un trascritto $\tau = (s^*, s_0, x_1, \ldots, x_\ell, L)$, dove L è la lista delle query effettuate a F. Un trascritto è *cattivo* se l'avversario interroga l'oracolo sullo stato iniziale s_0; tale evento ha probabilità al più $q/2^n$.

Nel mondo ideale ($b = 1$), la probabilità del trascritto segue la stessa forma del Lemma 16, con $p_1(\tau) = p_L \cdot 2^{-n}$, dove p_L rappresenta la coerenza delle query all'oracolo F. Per i trascritti buoni, un'analisi analoga alla *ratio analysis* del Lemma 16 mostra che

$$\frac{p_0(\tau)}{p_1(\tau)} \geq 1 - \frac{q\ell + \ell^2}{2^n}.$$

Poiché la struttura della prova ricalca esattamente quella illustrata nel **Lemma 18** (cfr. Sezione 6.4.2 [7]), se ne omettono qui i dettagli combinatori. Combinando i due contributi (con il Teorema sull'H–coefficient) si ottiene:

$$\mathrm{Adv}_{\mathrm{MD}}^{\mathrm{mnt},F}(\mathcal{A}) \leq \frac{q\ell + \ell^2}{2^n} + \frac{q}{2^n}.$$

La proprietà di maintaining security è quindi dimostrata.

Next Security – MD

La **next security** rappresenta la terza proprietà intermedia nella dimostrazione di robustezza. Essa richiede che l'uscita della funzione next, applicata a uno stato ad alta entropia (o già uniforme), sia indistinguibile da una stringa casuale di pari lunghezza. In altri termini, anche conoscendo lo stato interno, l'avversario non può prevedere l'uscita successiva del PRNG.

Come da **Lemma 19** del paper (cfr. Sezione 6.4.3 [7]), per ogni avversario (q)–legittimo vale:

$$\mathrm{Adv}_{\mathrm{MD}}^{\mathrm{next},F}(\mathcal{A}) \leq \varepsilon_{\mathrm{MD}}^{\mathrm{next}}(q) := \frac{q}{2^n}.$$

Dimostrazione. La prova segue direttamente la tecnica dell'H–coefficient, nella sua forma più semplice. Si consideri il trascritto

$$\tau = (s, s^*, L),$$

dove s è lo stato iniziale, s^* l'input fornito all'avversario e L la lista delle query a F. Un trascritto è *cattivo* se L contiene una query del tipo $((s, *), *) \in L$, ossia se l'avversario ha interrogato l'oracolo sullo stato corrente. In caso contrario, è detto *buono*.

Per i trascritti buoni, la distribuzione vista dall'avversario nei due mondi (reale e ideale) è identica, da cui $p_0(\tau) = p_1(\tau)$ e rapporto pari a 1. Nel mondo ideale, la probabilità che un trascritto sia cattivo è al più $q/2^n$, poiché ciascuna delle q query può colpire lo stato corrente s con probabilità 2^{-n}.

Applicando il teorema dell'H–coefficient, il vantaggio dell'avversario risulta quindi limitato da:

$$\mathrm{Adv}_{\mathrm{MD}}^{\mathrm{next},F}(\mathcal{A}) \leq \frac{q}{2^n}.$$

Recovering Security – MD

La **recovering security** assicura che, dopo un evento di perdita di entropia, una volta assorbiti nuovi input sufficientemente imprevedibili, lo stato e l'uscita del PRNG tornino indistinguibili da quelli di un generatore ideale. È quindi la proprietà che formalizza la capacità del PRNG di "recuperare" la propria casualità.

Come già menzionato, *questa proprietà si ottiene combinando le due proprietà intermedie di **extraction security** e **next security***, poiché l'estrazione di nuova entropia nello stato, seguita dall'applicazione della funzione `next`, garantisce che l'uscita finale risulti indistinguibile da una stringa uniforme.

Per ogni avversario (q, ℓ)–legittimo, vale:

$$\mathrm{Adv}_{\mathrm{MD}}^{\mathrm{rec},F}(\mathcal{A}) \leq \varepsilon_{\mathrm{MD}}^{\mathrm{ext}}(\gamma^*, q + \tfrac{r}{n} + 1, \ell) + \varepsilon_{\mathrm{MD}}^{\mathrm{next}}(q + \tfrac{r}{n} + 1)\,.$$

Dimostrazione. Si considerino due esperimenti H_0 e H_1 che corrispondono rispettivamente ai mondi reale e ideale del gioco di recovering, e un esperimento ibrido $H_{1/2}$ in cui il valore di sfida è generato applicando la funzione `next` a un input uniforme U_n. Per il teorema dell'H–coefficient e per la disuguaglianza triangolare, basta limitare la distanza tra H_0 e $H_{1/2}$ e tra $H_{1/2}$ e H_1.

La prima distanza è controllata dalla *extraction security* del PRNG: un avversario $\mathcal{A}_{\mathrm{ext}}$ può simulare l'esperimento reale interrogando l'oracolo F e inoltrando le proprie query alla sfida, ottenendo lo stesso transcript che in H_0. Quando il bit segreto del challenger vale 0, la simulazione coincide perfettamente con H_0; quando vale 1, l'avversario osserva un'istanza casuale di $next^F(U_n)$, che corrisponde alla distribuzione di $H_{1/2}$.

Pertanto, la distanza $\Delta(H_0, H_{1/2})$ è limitata da $\varepsilon_{\mathrm{MD}}^{\mathrm{ext}}(\gamma^*, q + r/n + 1, \ell)$.

In modo analogo, si costruisce un avversario $\mathcal{A}_{\mathrm{next}}$ che, partendo dall'ibrido $H_{1/2}$, simula il comportamento dell'avversario originale nell'esperimento ideale H_1. Qui la differenza fra i due mondi è controllata dalla proprietà di *next security*, da cui segue che $\Delta(H_{1/2}, H_1) \leq \varepsilon_{\mathrm{MD}}^{\mathrm{next}}(q + r/n + 1)$.

Applicando la disuguaglianza triangolare, la distanza complessiva tra H_0 e H_1 è al più la somma dei due contributi precedenti:

$$\mathrm{Adv}_{\mathrm{MD}}^{\mathrm{rec},F}(\mathcal{A}) \leq \varepsilon_{\mathrm{MD}}^{\mathrm{ext}}(\gamma^*, q + \tfrac{r}{n} + 1, \ell) + \varepsilon_{\mathrm{MD}}^{\mathrm{next}}(q + \tfrac{r}{n} + 1)\,.$$

Per una trattazione completa dei passaggi tecnici e delle costruzioni degli avversari $\mathcal{A}_{\mathrm{ext}}$ e $\mathcal{A}_{\mathrm{next}}$, si rimanda al **Lemma 20** in Sezione 6.4.4 [7].

Preserving Security – MD

La **preserving security** rappresenta l'ultima proprietà intermedia necessaria per dimostrare la robustezza del PRNG Merkle–Damgård. Essa garantisce che, partendo da uno stato ad alta entropia, l'assorbimento di nuovi input — anche scelti in modo avversario — non riduca la casualità dello stato interno né renda prevedibile l'uscita successiva del generatore.

Come già menzionato, *questa proprietà si ottiene combinando la **maintaining security** e la **next security***, poiché la capacità di mantenere l'imprevedibilità dello stato, unita all'indistinguibilità dell'uscita prodotta da next, assicura che l'intero processo resti sicuro anche in presenza di input potenzialmente controllati da un avversario.

Per ogni avversario (q, ℓ)–legittimo, vale:

$$\mathrm{Adv}^{\mathrm{pre},F}_{\mathrm{MD}}(\mathcal{A}) \le \varepsilon^{\mathrm{mnt}}_{\mathrm{MD}}(q + \tfrac{r}{n} + 1, \ell) + \varepsilon^{\mathrm{next}}_{\mathrm{MD}}(q + \tfrac{r}{n} + 1)\,.$$

Dimostrazione. La struttura della prova segue esattamente quella della *recovering security* (Lemma 20), sostituendo alla proprietà di *extraction security* quella di *maintaining security*. Si considerano due esperimenti H_0 e H_1 corrispondenti rispettivamente ai mondi reale e ideale del gioco di preservazione, e un esperimento ibrido $H_{1/2}$ in cui il valore di sfida è generato come $next^F(U_n)$, ossia applicando la funzione next a uno stato casuale uniforme.

La distanza tra H_0 e $H_{1/2}$ è limitata dalla *maintaining security* della costruzione MD: un avversario $\mathcal{A}_{\mathrm{mnt}}$ può simulare il comportamento di $\mathcal{A}$, inoltrando le query a F e riproducendo la stessa distribuzione del gioco reale. Quando il bit segreto vale 0, la simulazione è perfetta; quando vale 1, l'avversario osserva l'uscita di $next^F(U_n)$, che corrisponde all'ibrido $H_{1/2}$. Pertanto, $\Delta(H_0, H_{1/2}) \le \varepsilon^{\mathrm{mnt}}_{\mathrm{MD}}(q + r/n + 1, \ell)$.

Allo stesso modo, la distanza tra $H_{1/2}$ e H_1 è limitata dalla *next security*, poiché un avversario $\mathcal{A}_{\mathrm{next}}$ non può distinguere l'uscita prodotta da next da una stringa casuale uniforme. Segue quindi che $\Delta(H_{1/2}, H_1) \le \varepsilon^{\mathrm{next}}_{\mathrm{MD}}(q + r/n + 1)$.

Applicando la disuguaglianza triangolare, si ottiene la somma dei due contributi:

$$\mathrm{Adv}^{\mathrm{pre},F}_{\mathrm{MD}}(\mathcal{A}) \le \varepsilon^{\mathrm{mnt}}_{\mathrm{MD}}(q + \tfrac{r}{n} + 1, \ell) + \varepsilon^{\mathrm{next}}_{\mathrm{MD}}(q + \tfrac{r}{n} + 1)\,.$$

Per una trattazione completa dei passaggi tecnici e della costruzione degli avversari $\mathcal{A}_{\mathrm{mnt}}$ e $\mathcal{A}_{\mathrm{next}}$, si rimanda al **Lemma 21** in Sezione 6.4.5 [7].

2.6.5 PRNGs basati su Merkle–Damgård con Davies–Meyer

In questa sezione si dimostra la **robustezza** della costruzione di PRNG basata sul paradigma **Merkle–Damgård** con funzione di compressione **Davies–Meyer** (MD–DM), già presentata nella Sezione 2.5.2. La costruzione, indicata come $\mathsf{DM} = (\mathsf{refresh}, \mathsf{next})$, utilizza un cifrario ideale $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ e la sua definizione formale è riportata nella Costruzione 6.

L'obiettivo è mostrare che MD-DM risulta un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\mathrm{rob}})$–*robusto* nel modello E, con margine di sicurezza pari a:

$$\varepsilon_{\mathrm{rob}} \le 4t \cdot \left(\frac{\tilde{q}^{\,2} + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}} \right), \qquad \text{dove} \quad \tilde{q} = q + \frac{r}{n} + 1.$$

Come per la costruzione Merkle–Damgård basata su funzione casuale, la dimostrazione segue la tecnica dell'**H–coefficient** e si articola nelle cinque proprietà fondamentali: *extraction*, *maintaining*, *next*, *recovering* e *preserving security*. Si consiglia pertanto la lettura dell'altra costruzione per avere un'anteprima adeguata a questa sezione. La prova ricalca l'impostazione di Dodis et al. [7], adattandola al modello con cifrario ideale E e mostrando che le stesse relazioni di sicurezza valgono anche per la costruzione MD–DM.

Extraction Security – MD-DM

La prima proprietà da verificare per la costruzione Merkle–Damgård con funzione di compressione **Davies–Meyer** (MD–DM) è la **extraction security**. Essa garantisce che, dopo l'assorbimento di blocchi di input ad alta entropia, lo stato interno del PRNG diventa indistinguibile da una distribuzione uniforme, anche per un avversario che osservi le query effettuate al cifrario ideale E.

Secondo il **Lemma 22** del paper (cfr. Sezione 6.5.1. [7]), per ogni avversario (q, ℓ)–legittimo vale:

$$\mathrm{Adv}_{\mathrm{DM}}^{\mathrm{ext},E}(\mathcal{A}) \leq \varepsilon_{\mathrm{MD}}^{\mathrm{ext}}(\gamma^*, q, \ell) := \frac{q^2 + 2(q\ell + \ell^2)}{2^n} + \frac{4q}{2^{\gamma^*}}.$$

Setup e notazioni. Come nel caso Merkle–Damgård con funzione casuale, consideriamo un trascritto

$$\tau = (s^*, s_0, x_1, \ldots, x_\ell, L),$$

che descrive l'interazione tra l'avversario $\mathcal{A}$ e l'oracolo ideale E. Qui s_0 è lo stato iniziale, s^* è l'output finale del gioco, e L rappresenta l'insieme delle query fatte all'oracolo E.

Rispetto alla costruzione MD classica, le query hanno ora la forma

$((x_i, y_{i-1}),\ y_i \oplus y_{i-1}) \in L$, poiché nella variante Davies–Meyer la compressione è definita come $y_i = E(x_i, y_{i-1}) \oplus y_{i-1}$. Sia $\ell' \geq 0$ il più grande indice tale che di questa forma risultino verificate per alcuni valori $y_0, y_1, \ldots, y_{\ell'}$, con $y_0 = s_0$.

Un trascritto τ è detto *cattivo* se:

- $\ell' = \ell$, oppure
- $\ell' = \ell - 1$ e se esiste una $((x_\ell, *),\ y_{\ell-1} \oplus s^*) \in L$.

In altre parole, se l'avversario ha interrogato il cifrario E su tutte le coppie necessarie a ricostruire lo stato finale s^*, oppure se ha posto una query che ne rivela indirettamente il valore tramite l'operazione di XOR con l'ultimo chaining value $y_{\ell-1}$. In entrambi i casi, lo stato finale non può più essere considerato "fresco", poiché le query forniscono informazioni sufficienti a determinarlo. In tutti gli altri casi, τ è *buono*.

Come nel caso Merkle–Damgård, l'obiettivo è stimare la distanza statistica tra il mondo reale e quello ideale mediante la tecnica dell'H–coefficient, limitando il rapporto delle probabilità per i trascritti buoni e la probabilità di occorrenza dei trascritti cattivi.

Analisi del Rapporto. Ricordiamo che per ogni trascritto buono τ, vale il seguente bound:

$$\frac{p_0(\tau)}{p_1(\tau)} \geq 1 - \frac{2(q\ell + \ell^2)}{2^n}.$$

Dimostrazione. Come nel caso Merkle–Damgård (v. Sezione 2.6.4), si fissi un trascritto buono τ e si considerino le due probabilità nei mondi reale e ideale. Nel mondo ideale, l'uscita s^* è campionata uniformemente, per cui

$$p_1(\tau) = p_L \cdot 2^{-n},$$

dove p_L è la probabilità che un cifrario ideale casuale E risulti coerente con l'insieme di query L. Nel mondo reale, invece,

$$p_0(\tau) = p_L \cdot q_\tau, \qquad q_\tau = \Pr\big[\mathsf{MD\text{–}DM}_{s_0}^{E_L}(x_1, \ldots, x_\ell) = s^*\big].$$

Il termine q_τ rappresenta la probabilità che la catena Merkle–Damgård–Davies–Meyer, calcolata a partire da s_0 e condizionata alle query già osservate, produca esattamente s^*.

Poiché τ è buono, esiste un indice $\ell' < \ell$ tale che gli stati intermedi $y_{\ell'+1}, \ldots, y_\ell$ non siano determinati dalle query viste. Si consideri quindi

$$Y_\ell := \mathsf{DM}_{y_{\ell'}}^{E_L}(x_{\ell'+1}, \ldots, x_\ell) = s^*.$$

Per stimare q_τ è necessario analizzare i possibili casi in cui la catena è parzialmente determinata:

- **Caso (1):** $\ell' \leq \ell - 2$. Almeno due passi della catena rimangono "non visti" dall'avversario; in questo scenario si applica un'analisi analoga al caso Merkle–Damgård classico.
- **Caso (2):** $\ell' = \ell - 1$. Solo l'ultimo blocco non è determinato da L, e s^* dipende direttamente da una singola applicazione del cifrario E.

In entrambi i casi si ottiene lo stesso tipo di bound.

Sia ora L_i l'insieme delle query al cifrario fino al passo i, definito come

$$L_i := L_{i-1} \cup \{((x_i, Y_{i-1}),\, Y_i \oplus Y_{i-1})\}, \quad L_{\ell'} := L.$$

Per una collezione di query $\tilde{L}$, indichiamo con $\mathsf{Free\text{-}In}_L$ e $\mathsf{Free\text{-}Out}_L$ i rispettivi insiemi di input e output "liberi", cioè non ancora utilizzati dal cifrario E:

$$y \in \mathsf{Free\text{-}In}_{\tilde{L}} \iff ((*, y), *) \notin \tilde{L}, \qquad y \in \mathsf{Free\text{-}Out}_{\tilde{L}} \iff ((*, *), y) \notin \tilde{L}.$$

Definiamo quindi, per ogni passo $i = \ell' + 1, \ldots, \ell - 2$, l'evento di freschezza:

$$\mathsf{FRESH}_i := (Y_i \in \mathsf{Free\text{-}In}_{L_{i-1}}),$$

e, per l'ultimo passo utile, richiediamo inoltre che l'output XORato sia ancora libero:

$$Y_{\ell-1} \oplus s^* \in \mathsf{Free\text{-}Out}_{L_{\ell-2}}.$$

Infine,

$$\mathsf{FRESH} := \bigcap_{i=\ell'+1}^{\ell-1} \mathsf{FRESH}_i.$$

Condizionato a FRESH, l'ultimo stato Y_ℓ è campionato uniformemente, poiché E si comporta come un cifrario casuale indipendente:

$$\Pr[Y_\ell = s^* \mid \mathsf{FRESH}] = 2^{-n}.$$

Rimane ora da valutare la probabilità che l'evento FRESH si verifichi. Un vincolo di "non freschezza" si verifica se, durante la costruzione, viene effettuata una query che coinvolge un input o output già usato. Ad esempio, per valori compatibili con il conditioning, si ha violazione di freschezza quando

$$E_L(x_{\ell-1}, y_{\ell-2}) \in \overline{\mathsf{Free\text{-}In}_{L_{\ell-2}}} \oplus y_{\ell-2}$$

oppure

$$E_L(x_{\ell-1}, y_{\ell-2}) \in \overline{\mathsf{Free\text{-}Out}_{L_{\ell-2}}} \oplus y_{\ell-2} \oplus s^*.$$

La probabilità di tali collisioni può essere limitata come:

$$\frac{|\mathsf{Free\text{-}Out}_{L_{\ell-2}} \cap \overline{\mathsf{Free\text{-}In}_{L_{\ell-2}}} \oplus y_{\ell-2}|}{|\mathsf{Free\text{-}Out}_{L_{\ell-2}}|} \leq \frac{q+\ell}{2^n - (q+\ell)}.$$

Un argomento analogo fornisce lo stesso bound per l'insieme complementare degli eventi di violazione, da cui segue che:

$$\Pr\left[\bigcap_{k=\ell'+1}^{i-1} \mathsf{FRESH}_k\right] \geq 1 - \frac{q\ell + \ell^2}{2^n - (q+\ell)}.$$

(Nella trattazione originale vengono esplicitate le relazioni tra gli insiemi $\mathsf{Free\text{-}In}$ *e* $\mathsf{Free\text{-}Out}$*, oltre a una serie di disuguaglianze combinatorie intermedie; tali passaggi tecnici sono omessi per brevità, poiché conducono allo stesso bound finale).*

Combinando i risultati precedenti si ottiene:

$$\begin{aligned} q_\tau = \Pr[Y_\ell = s^*] &\geq \Pr[\mathsf{FRESH}] \cdot \Pr[Y_\ell = s^* \mid \mathsf{FRESH}] \\ &\geq \left(1 - \frac{q\ell + \ell^2}{2^n - (q+\ell)}\right) \cdot 2^{-n}. \end{aligned}$$

Infine, sostituendo le espressioni di $p_0(\tau)$ e $p_1(\tau)$ si ottiene il bound complessivo:

$$\frac{p_0(\tau)}{p_1(\tau)} = \frac{q_\tau}{2^{-n}} \geq 1 - \frac{2(q\ell + \ell^2)}{2^n}.$$

Questo risultato mostra che, per tutti i trascritti buoni, le due distribuzioni (reale e ideale) differiscono solo di una quantità trascurabile, proporzionale alla probabilità di collisione tra le query al cifrario E. Il fattore 2 additivo rispetto al caso Merkle–Damgård riflette la maggiore possibilità di collisioni introdotta dall'operazione XOR nella compressione di Davies–Meyer.

Per una dimostrazione completa, si veda il **Lemma 23** in Sezione 6.5.1 [7].

Analisi degli Eventi Rari. Analogamente al caso Merkle–Damgård, si considera ora la probabilità che un trascritto τ risulti *cattivo*, ossia che l'avversario abbia interrogato il cifrario E su tutte le coppie necessarie a ricostruire lo stato finale s^*, oppure che esista una query che ne riveli indirettamente il valore tramite XOR. Nel modello ideale ($b = 1$), l'uscita s^* è indipendente dallo stato iniziale e dagli input forniti, e l'analisi procede mediante una tecnica di *resampling* analoga a quella utilizzata nella prova del Lemma 17 per la costruzione MD (v. Sezione 2.6.4).

Il **Lemma 24** del paper (cfr. Sezione 6.5.1. [7]) stabilisce che la probabilità di ottenere un trascritto cattivo è limitata da:

$$\Pr[T_1 \in \mathcal{B}] \leq \frac{4q}{2^{\gamma^*}} + \frac{q^2}{2^n}.$$

L'idea principale della dimostrazione è che le collisioni nel modello di cifrario richiedono un'attenzione aggiuntiva rispetto al caso di funzione casuale: due query $E((k,u),v)$ e $E((k',u'),v')$ si dicono in collisione se

$$v \oplus u = v' \oplus u'.$$

Si definisce $\mathcal{E}$ come l'evento che due query in $\mathcal{L}$ siano in collisione. Come nel caso precedente,

$$\Pr[T_1 \in \mathcal{B}] \leq \Pr[T_1 \in \mathcal{B} \mid \mathcal{E}] + \Pr[\mathcal{E}] \leq \Pr[T_1 \in \mathcal{B} \mid \mathcal{E}] + \frac{q^2}{2^n}.$$

Per limitare il primo termine, si considerano le possibili *catene potenziali* (*potential chains*) che collegano le query dell'avversario. Sia $z = (\sigma, L, s_0)$ una configurazione collision-free, e sia Z la variabile casuale corrispondente. Condizionato su $Z = z$, il gioco produce un trascritto cattivo solo se l'avversario riesce a ricostruire una delle catene ammesse con le relazioni:

$$((k_i, y_{i-1}), y_i \oplus y_{i-1}) \in L, \quad i = 1, \ldots, \ell,$$

oppure, per l'ultimo passo,

$$((k_\ell, y_{\ell-1}), y_{\ell-1} \oplus s^*) \in L.$$

Senza collisioni, l'insieme L può contenere al più $2q$ catene potenziali. Ogni nuova query può estendere una catena esistente o crearne una nuova, ma al massimo due nuove catene possono essere introdotte per volta. Ne consegue che, condizionato su $Z = z$, il vantaggio dell'avversario è limitato da:

$$\Pr[T_1 \in \mathcal{B} \mid Z = z] \leq 2q \cdot p_z, \qquad \textit{dove} \;\; p_z := \mathrm{Pred}(\overline{X} \mid Z = z).$$

In media, rispetto alla distribuzione di Z, si ottiene:

$$\Pr[T_1 \in \mathcal{B} \mid \mathcal{E}] \leq 2q \cdot \mathrm{Pred}(\overline{X} \mid Z\mathcal{E}) \leq \frac{4q}{2^{\gamma^*}},$$

dove l'ultima disuguaglianza segue dal vincolo di min-entropia della sorgente. Combinando i risultati si ottiene quindi il bound finale:

$$\Pr[T_1 \in \mathcal{B}] \leq \frac{4q}{2^{\gamma^*}} + \frac{q^2}{2^n}.$$

(Nella trattazione originale vengono inoltre esplicitati i casi di collisione tra query e le condizioni di indipendenza tra catene potenziali; tali dettagli combinatori, del tutto analoghi a quelli discussi per la costruzione MD, sono qui omessi per brevità).

Maintaining Security – MD-DM

La proprietà di **maintaining security** per la costruzione Merkle–Damgård con schema Davies–Meyer (MD–DM) viene dimostrata con la stessa tecnica H–coefficient utilizzata nel caso Merkle–Damgård classico (v. Sezione 2.6.4), tenendo conto delle differenze introdotte dall'uso di un cifrario ideale E.

Il **Lemma 25** del paper (cfr. Sezione 6.5.2. [7]) stabilisce che, per ogni avversario (q, ℓ)–legittimo, il vantaggio nel distinguere i due mondi (reale e ideale) è limitato da:

$$\mathrm{Adv}_{\mathrm{DM}}^{\mathrm{mnt},E}(\mathcal{A}) \leq \varepsilon_{\mathrm{DM}}^{\mathrm{mnt}}(q,\ell) := \frac{2(q\ell+\ell^2)}{2^n} + \frac{q}{2^n}.$$

Dimostrazione. Analogamente al caso precedente, si considera un trascritto di interazione

$$\tau = (s^*, s_0, x_1, \ldots, x_\ell, L),$$

dove s^* rappresenta il valore finale restituito dal gioco e L la lista di query effettuate dall'avversario al cifrario E. Un trascritto τ è *cattivo* se L contiene una query della forma $((*, s_0), *)$, ovvero se l'avversario ha interrogato E utilizzando lo stato iniziale s_0. Nel mondo ideale ($b = 1$), la probabilità che ciò avvenga è al più $|L|/2^n$.

Per i trascritti buoni, valgono le stesse relazioni viste nel caso MD:

$$p_1(\tau) = 2^{-2n} \cdot p_L, \qquad p_0(\tau) \geq \Big(1 - \frac{2(q\ell+\ell^2)}{2^n}\Big) \cdot 2^{-2n} \cdot p_L,$$

dove p_L denota la probabilità che un cifrario casuale uniforme risulti coerente con le query in L. Il termine aggiuntivo nel numeratore riflette la possibile duplicazione degli argomenti dovuta alla struttura Davies–Meyer, dove le collisioni possono avvenire sia sull'input u che sull'output $u \oplus E(k, u)$.

La derivazione formale segue passo per passo quella della *ratio analysis* nel Lemma 23, e porta al bound finale sopra riportato.

Next Security – MD-DM

La proprietà di **next security** per la costruzione Merkle–Damgård con schema Davies–Meyer (MD–DM) si dimostra in modo analogo al caso Merkle–Damgård classico (v. Sezione 2.6.4), utilizzando la tecnica dell'H–coefficient per confrontare il comportamento del cifrario ideale E con quello di un oracolo casuale.

Il **Lemma 26** del paper(cfr. Sezione 6.5.3. [7]) afferma che il vantaggio di un avversario $\mathcal{A}$ nel distinguere l'uscita della funzione `next` da una stringa uniforme è limitato da:

$$\mathrm{Adv}_{\mathrm{DM}}^{\mathrm{next},E}(\mathcal{A}) \leq \varepsilon_{\mathrm{DM}}^{\mathrm{next}}(q) := \frac{q}{2^n}.$$

Dimostrazione. La prova segue direttamente l'impostazione del caso Merkle–Damgård, considerando un trascritto di interazione

$$\tau = (s, s^*, s_0^*, s_1^*, \ldots, s_{r/n}^*, L),$$

dove s è lo stato iniziale, gli elementi s_i^* rappresentano le risposte fornite al challenger, e L è la lista delle query effettuate dall'avversario al cifrario E. Un trascritto è *cattivo*

se L contiene una query della forma $((i, s), *)$ oppure $((i, *), s \oplus s_i^*)$ per qualche i; altrimenti è detto *buono*.

Nel caso di trascritti buoni, il rapporto delle probabilità nei due mondi è almeno 1, poiché la distribuzione delle risposte di E risulta perfettamente uniforme rispetto allo stato s. Nel mondo ideale, invece, in cui la vista di $\mathcal{A}$ è completamente indipendente da s, la probabilità di un evento cattivo è limitata da:

$$\Pr[T_1 \in \mathcal{B}] \leq \frac{q}{2^n}.$$

Questo porta immediatamente al bound desiderato sul vantaggio dell'avversario:

$$\mathrm{Adv}_{\mathrm{DM}}^{\mathrm{next},E}(\mathcal{A}) \leq \frac{q}{2^n}.$$

Recovering Security – MD-DM

In questa sezione si combinano le proprietà di *extraction* e *next security* per ottenere la **recovering security** della costruzione Merkle–Damgård con schema Davies–Meyer (MD–DM). Sia $\varepsilon_{\mathrm{DM}}^{\mathrm{ext}}(\gamma^*, q, \ell)$ e $\varepsilon_{\mathrm{DM}}^{\mathrm{next}}(q)$ come nei Lemmi 22 e 26, rispettivamente.

Il **Lemma 27** del paper (cfr. Sezione 6.5.4. [7]) stabilisce che, per ogni avversario (q, ℓ)–legittimo, vale:

$$\mathrm{Adv}_{\mathrm{DM}}^{\mathrm{rec},E}(\mathcal{A}) \leq \varepsilon_{\mathrm{DM}}^{\mathrm{ext}}(\gamma^*, \ell, q + r/n + 1) + \varepsilon_{\mathrm{DM}}^{\mathrm{next}}(q + r/n + 1).$$

La dimostrazione segue lo stesso schema del Lemma 20 (v. Sezione 2.6.4), relativo alla costruzione Merkle–Damgård.

Preserving Security – MD-DM

In modo analogo, combinando le proprietà di *maintaining* e *next security* si ottiene la **preserving security** per la costruzione MD–DM. Sia $\varepsilon_{\mathrm{DM}}^{\mathrm{mnt}}(q, \ell)$ e $\varepsilon_{\mathrm{DM}}^{\mathrm{next}}(q)$ come nei Lemmi 25 e 26.

Il **Lemma 28** del paper (cfr. Sezione 6.5.5. [7]) afferma che, per ogni avversario $\mathcal{A}$, vale:

$$\mathrm{Adv}_{\mathrm{DM}}^{\mathrm{pre},E}(\mathcal{A}) \leq \varepsilon_{\mathrm{DM}}^{\mathrm{mnt}}(q + r/n + 1, \ell) + \varepsilon_{\mathrm{DM}}^{\mathrm{next}}(q + r/n + 1).$$

Anche in questo caso, la dimostrazione è del tutto analoga a quella del Lemma 21 (v. Sezione 2.6.4).

2.6.6 PRNGs basati su Sponge

In questa sezione si dimostra la **robustezza** della costruzione di PRNG basata sul paradigma **Sponge**, già presentata nella Sezione 2.5.3. Si ricordi che la costruzione $\mathsf{Spg} = (\mathsf{refresh}, \mathsf{next})$ utilizza una permutazione ideale $\pi : \{0,1\}^n \to \{0,1\}^n$, e che la sua definizione formale è riportata in Costruzione 8.

L'obiettivo di questa sezione è mostrare che tale costruzione risulta un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\mathrm{rob}})$-robusto nel modello π, con margine di sicurezza pari a:

$$\varepsilon_{\mathrm{rob}} \leq 4t \cdot \left(\frac{\tilde{q}^{\,2} + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}} + \frac{\tilde{q}^2}{2^c} \right), \qquad \text{dove} \quad \tilde{q} = q + \frac{r}{n} + 1.$$

Si consiglia la lettura dell'altra costruzione per avere un'anteprima adeguata a questa sezione, poiché la dimostrazione della robustezza della costruzione Sponge segue lo stesso schema logico della costruzione Merkle–Damgård, con alcune differenze dovute alla natura della permutazione π. In particolare, mentre le costruzioni MD e MD–DM soddisfano direttamente le proprietà di *extraction* e *maintaining security*, nel caso Sponge la prova introduce il concetto di **hit probability**, che rappresenta la probabilità che l'avversario interroghi π^{-1} sul valore finale dello stato interno. Tale probabilità è trascurabile nel mondo ideale ($b = 1$), e la sua gestione permette di estendere i risultati di *recovering* e *preserving security* anche a questa costruzione.

Extraction Security – SP

La prima proprietà di sicurezza da verificare per la costruzione Sponge è la **extraction security**, che garantisce che, dopo l'assorbimento di blocchi di input con sufficiente entropia, lo stato del PRNG risulti indistinguibile da una distribuzione uniforme. Come nelle costruzioni precedenti, l'obiettivo è mostrare che lo stato ottenuto dopo le fasi di assorbimento mantiene una distribuzione statisticamente indistinguibile da una stringa casuale, anche in presenza di un avversario che interagisca con la permutazione ideale π.

Il **Lemma 29** del paper (cfr. Sezione 6.6.1. [7]) stabilisce che, per ogni avversario γ^*–legittimo (q, ℓ), vale il seguente bound sul vantaggio di $\mathcal{A}$:

$$\mathrm{Adv}^{\mathrm{ext},\pi}_{\mathrm{Spg}}(\mathcal{A}) \leq \varepsilon^{\mathrm{ext}}_{\mathrm{Spg}}(\gamma^*, \ell, q) := \frac{q + 2(q\ell + \ell^2)}{2^n} + \frac{2q}{2^{\gamma^*}} + \frac{q^2}{2^c} + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A}),$$

dove $\mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A})$ rappresenta la probabilità che l'avversario effettui una query inversa a π sul valore finale s^* prodotto dal gioco.

Per comodità, si definisce inoltre:

$$\delta^{\mathrm{ext}}_{\mathrm{Spg}}(\gamma^*, \ell, q) := \frac{q + 2(q\ell + \ell^2)}{2^n} + \frac{2q}{2^{\gamma^*}} + \frac{q^2}{2^c},$$

così che il vantaggio totale risulti:

$$\mathrm{Adv}^{\mathrm{ext},\pi}_{\mathrm{Spg}}(\mathcal{A}) \leq \delta^{\mathrm{ext}}_{\mathrm{Spg}}(\gamma^*, \ell, q) + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A}).$$

Setup e notazioni. Si considera un trascritto

$$\tau = (s^*, s_0, x_1, \ldots, x_\ell, L),$$

che rappresenta l'interazione tra l'avversario $\mathcal{A}$ e l'oracolo π. L'insieme L raccoglie tutte le coppie di query (u, v) effettuate da $\mathcal{A}$, dove $v = \pi(u)$. Rispetto ai casi Merkle–Damgård e Merkle–Damgård con Davies–Meyer, le query hanno ora la forma $(u_i, v_i) \in L$, dove $u_1 = s_0 \oplus x_1 \| 0^c$, e con

$$u_i = v_{i-1} \oplus x_i \| 0^c \quad \text{per} \quad i = 2, \ldots, \ell - 1.$$

Sia $\ell' \geq 0$ il massimo indice tale che queste condizioni valgano per alcuni $u_1, \ldots, u_{\ell'}$, mentre per $i > \ell'$ il calcolo prosegue senza che le corrispondenti coppie (u_i, v_i) appartengano a L.

Un trascritto τ è detto *cattivo* se si verifica una delle seguenti condizioni:

- **(hit)** — se $(*, s^*) \in L$, ossia l'avversario ha interrogato direttamente l'inversa π^{-1} sul valore finale s^*;
- **(chain)** — se le query ricostruiscono completamente la catena di stati, cioè $\ell' = \ell$.

Nel primo caso si verifica un *hit event*, mentre nel secondo il trascritto è completo e quindi potenzialmente rivelante.

Come nelle costruzioni MD e MD–DM, la prova si suddivide in due parti: l'analisi del rapporto tra i mondi reale e ideale (Lemma 30) e la stima della probabilità di trascritti cattivi (Lemma 31). La differenza principale risiede nella gestione dei casi "**hit**", che richiedono di introdurre il termine addizionale $\mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A})$ nel bound finale.

Analisi del Rapporto. Ricordiamo che per ogni trascritto buono τ, vale

$$\frac{p_0(\tau)}{p_1(\tau)} \geq \left(1 - \frac{2(q\ell + \ell^2)}{2^n}\right).$$

Dimostrazione. Come nei casi Merkle–Damgård e Merkle–Damgård–Davies–Meyer, si considerano le due probabilità associate a τ nei mondi reale e ideale:

$$p_1(\tau) = p_L \cdot 2^{-n}, \qquad p_0(\tau) = p_L \cdot q_\tau,$$

dove p_L è la probabilità che una permutazione casuale π risulti coerente con le query registrate in L, e

$$q_\tau = \Pr\left[\mathsf{Sponge}^{\pi_L}_{s_0}(x_1, x_2, \ldots, x_\ell) = s^*\right].$$

In questo caso, il termine q_τ rappresenta la probabilità che, dato il comportamento della permutazione π sulle query già osservate, la costruzione produca esattamente lo stato finale s^*. In altre parole, quantifica quanto la sequenza reale di calcoli si discosta da una permutazione perfettamente casuale.

Poiché τ è buono, la catena interna non è determinata interamente da L (ossia $\ell' < \ell$). Si ha quindi che q_τ è la probabilità, rispetto alla scelta di π_L, che

$$V_\ell := \mathsf{Sponge}^{\pi_L}_{v_{\ell'}}(x_{\ell'+1}, \ldots, x_\ell) = s^*.$$

Consideriamo gli stati intermedi $U_{\ell'+1}, V_{\ell'+1}, \ldots, U_{\ell-1}, V_{\ell-1}, U_\ell, V_\ell$, definiti da:

$$U_{\ell'+1} = v_{\ell'} \oplus (x_{\ell'+1} \| 0^c), \qquad U_i = V_{i-1} \oplus (x_i \| 0^c), \qquad V_i = \pi(U_i),$$

per $i = \ell' + 2, \ldots, \ell$.

Per $i = \ell' + 1, \ldots, \ell$, si definisce l'evento di freschezza FRESH_i:

- U_i è *fresh*, cioè non compare come input in L e $U_i \neq U_j$ per $j < i$;
- V_{i-1} non è un *hit*, ossia $V_{i-1} \neq s^*$.

L'evento complessivo di freschezza è quindi

$$\mathsf{FRESH} := \bigcap_{i=\ell'+1}^{\ell} \mathsf{FRESH}_i.$$

Intuitivamente, la condizione FRESH garantisce che gli input forniti alla permutazione siano nuovi e indipendenti dalle query precedenti, permettendo di trattare i risultati come valori uniformemente distribuiti.

Condizionato su FRESH, l'ultimo stato è distribuito uniformemente, perciò

$$\Pr[V_\ell = s^* \mid \mathsf{FRESH}] = 2^{-n}.$$

Per stimare la probabilità di freschezza, si osserva che ad ogni passo possono esistere al più $q + \ell$ valori già utilizzati (come input o output della permutazione); di conseguenza:

$$\Pr[\mathsf{FRESH}] \geq 1 - \frac{q\ell + \ell^2}{2^n - (q - \ell)}.$$

Il termine corretto al denominatore riflette il fatto che il numero di elementi "non freschi" riduce la dimensione effettiva dello spazio su cui π agisce. Trascurando termini di ordine inferiore, si ottiene la stima approssimata

$$\Pr[\mathsf{FRESH}] \geq 1 - \frac{2(q\ell + \ell^2)}{2^n}.$$

Combinando i risultati precedenti,

$$\begin{aligned} q_\tau = \Pr[V_\ell = s^*] &= \Pr[\mathsf{FRESH}] \cdot \Pr[V_\ell = s^* \mid \mathsf{FRESH}] \\ &\geq \left(1 - \frac{2(q\ell + \ell^2)}{2^n}\right) \cdot 2^{-n}. \end{aligned}$$

Infine,

$$\frac{p_0(\tau)}{p_1(\tau)} = \frac{q_\tau}{2^{-n}} \geq 1 - \frac{2(q\ell + \ell^2)}{2^n}.$$

Questo risultato mostra che, per tutti i trascritti buoni, le distribuzioni nei mondi reale e ideale differiscono solo di una quantità trascurabile, proporzionale alla probabilità di collisione tra le query alla permutazione π. Il fattore 2 nel bound finale riflette la possibilità che collisioni o eventi di tipo *hit* si verifichino in due direzioni (input e output), caratteristica distintiva del paradigma Sponge. Per una trattazione completa, si veda il **Lemma 30** in Sezione 6.6.1 di [7].

Analisi degli Eventi Rari. Analogamente ai casi precedenti, consideriamo ora la probabilità che un trascritto τ sia *cattivo*. In questo contesto, un trascritto è detto cattivo se si verifica uno dei seguenti eventi:

- si presenta un *hit*, ovvero una query diretta o inversa alla permutazione π restituisce il valore finale s^*;

- si verifica una collisione tra query che condividono lo stesso valore di capacità, compromettendo la proprietà di indipendenza delle valutazioni di π.

Il **Lemma 31** del paper (cfr. Sezione 6.6.1. [7]) stabilisce che:

$$\Pr[T_1 \in \mathcal{B}] \leq \frac{2q}{2^{\gamma^*}} + \frac{q}{2^n} + \frac{q^2}{2^c} + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A}),$$

dove $\mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A})$ è il vantaggio dell'avversario nel formulare una query inversa su s^*.

Dimostrazione. La strategia di dimostrazione riprende la tecnica di *resampling* già vista nel Lemma 17. Un evento di tipo *hit* si verifica quando una query diretta o inversa produce esattamente il valore s^*. La probabilità complessiva di tale evento è quindi limitata da:

$$\frac{q}{2^n} + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A}).$$

Per analizzare invece le collisioni, si costruisce un grafo diretto $G = (V, E)$ i cui nodi rappresentano le *capacity parts* dei valori in L, ovvero:

$$V = \{v^{(c)} \mid (v, *) \in L \text{ oppure } (*, v) \in L\}.$$

Due nodi sono connessi da un arco etichettato se la corrispondente query è stata eseguita:

$$E = \{(u^{(c)}, v^{(c)}, l) \mid (u, v) \in L,\ l = u^{(r)} \oplus v^{(r)}\}.$$

L'insieme L è detto *collision–free* se, per ogni coppia di nodi, esiste al più un cammino con etichette univoche che li collega nel grafo G. Sia $\mathcal{E}$ l'evento in cui L non è collision–free.

Per decomporre i casi,

$$\Pr[T_1 \in \mathcal{B}] \leq \Pr[T_1 \in \mathcal{B} \mid \mathcal{E}] + \Pr[\bar{\mathcal{E}}] \leq \Pr[T_1 \in \mathcal{B} \mid \mathcal{E}] + \frac{q^2}{2^c}.$$

L'ultimo termine riflette la probabilità che due query alla permutazione π collidano nella parte di capacità.

Per limitare $\Pr[T_1 \in \mathcal{B} \mid \mathcal{E}]$, si considera una tripla $z = (\sigma, L, s_0) \in \mathcal{E}$ e si definisce una *potential chain* come una sequenza $(u_1, v_1), \ldots, (u_\ell, v_\ell) \in L$ tale che

$$u_1 = s_0 \oplus \mu_1 \| 0^c, \qquad u_i = v_{i-1} \oplus \mu_i \| 0^c, \quad i = 2, \ldots, \ell - 1.$$

I valori $\mu_1, \ldots, \mu_\ell$ rappresentano le etichette che descrivono il percorso da $s_0^{(c)}$ a $v_\ell^{(c)}$ nel grafo G. In un insieme collision–free, possono esistere al più q catene potenziali di questo tipo.

Condizionando su $Z = z$, il gioco T_1 produce un trascritto cattivo se e solo se per una di tali catene $X_i = \mu_i$ per ogni $i = 1, \ldots, \ell$. Per la legittimità dell'avversario, segue che

$$\Pr[T_1 \in \mathcal{B} \mid Z = z] \leq q \cdot p_z, \qquad p_z := \mathrm{Pred}(\overline{X} \mid Z = z).$$

Combinando i risultati precedenti si ottiene il bound complessivo:

$$\Pr[T_1 \in \mathcal{B}] \leq \frac{2q}{2^{\gamma^*}} + \frac{q}{2^n} + \frac{q^2}{2^c} + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A}),$$

Maintaining Security – SP

La proprietà di **maintaining security** per la costruzione Sponge–PRNG si dimostra con la stessa tecnica H–coefficient adottata per MD e MD–DM, tenendo conto dell'uso di una *permutazione ideale* π e del possibile *hit* sull'uscita finale tramite query diretta o inversa.

Il **Lemma 32** del paper (cfr. Sezione 6.6.2 [7]) stabilisce che, per ogni avversario (q, ℓ)–legittimo, il vantaggio nel distinguere i due mondi (reale e ideale) è limitato da:

$$\mathrm{Adv}^{\mathrm{mntn},\pi}_{\mathrm{Spg}}(\mathcal{A}) \leq \varepsilon^{\mathrm{mntn}}_{\mathrm{Spg}}(q,\ell) := \frac{2(q\ell+\ell^2)}{2^n} + \frac{2q}{2^n} + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A})\,.$$

Dimostrazione. Consideriamo un trascritto di interazione

$$\tau = (s^*, s_0, x_1, \ldots, x_\ell, L),$$

dove s^* è il valore restituito dal gioco e L è l'insieme delle query di $\mathcal{A}$ alla permutazione π. Un trascritto τ è *cattivo* se L contiene una query del tipo:

- $(s_0, *)$: nel mondo ideale ($b = 1$) avviene con probabilità al più $|L|/2^n$, poiché s_0 è indipendente dalla vista di $\mathcal{A}$;
- $(*, s^*)$: può avvenire via query diretta (probabilità $|L|/2^n$) oppure via *backward query* su s^*, catturata dal termine $\mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A})$.

Per i trascritti *buoni* (cioè privi delle query sopra), valgono, come nel caso dell'*extraction* (v. Lemma 30 2.6.6),

$$p_1(\tau) = 2^{-2n}\,p_L, \qquad p_0(\tau) \;\geq\; \Big(1 - \frac{2(q\ell+\ell^2)}{2^n}\Big)\,2^{-2n}\,p_L,$$

dove p_L è la probabilità che una permutazione uniforme risulti coerente con le query in L. Ne segue immediatamente il rapporto:

$$\frac{p_0(\tau)}{p_1(\tau)} \;\geq\; 1 - \frac{2(q\ell+\ell^2)}{2^n}.$$

Combinando il contributo dei trascritti buoni con la probabilità dei trascritti cattivi (si sommano i termini $|L|/2^n$ relativi agli hit su s_0 e su s^*, più l'eventuale query inversa su s^* assorbita in $\mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A})$), si ottiene il bound complessivo enunciato:

$$\mathrm{Adv}^{\mathrm{mtn},\pi}_{\mathrm{Spg}}(\mathcal{A}) \leq \frac{2(q\ell+\ell^2)}{2^n} + \frac{2q}{2^n} + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A}).$$

Next Security – SP

La proprietà di **next security** per la costruzione Sponge–PRNG assicura che l'uscita prodotta dalla funzione next, quando applicata a uno stato interno ad alta entropia, risulti indistinguibile da una stringa casuale uniforme. In altri termini, anche se l'avversario effettua fino a q query alla permutazione ideale π, non può distinguere l'output generato da next da un valore uniforme in $\{0,1\}^{n+r}$.

Il **Lemma 33** del paper (cfr. Sezione 6.6.3 [7]) stabilisce il limite al vantaggio di un avversario contro la proprietà di **next security** nella costruzione Sponge:

$$\mathrm{Adv}^{\mathrm{next},\pi}_{\mathrm{Spg}}(\mathcal{A}) \leq \varepsilon^{\mathrm{next}}_{\mathrm{Spg}}(q) := \frac{2q}{2^c}.$$

Dimostrazione. Si considera un trascritto

$$\tau = (s_0^{(c)},\, s^*,\, y^*,\, L),$$

dove $s_0^{(c)}$ è la parte di capacità dello stato iniziale, s^* il nuovo stato e y^* l'uscita prodotta da next, mentre L rappresenta le query dell'avversario $\mathcal{A}$ alla permutazione π.

Un trascritto τ è *cattivo* se L contiene una query del tipo:

- $(y^* \,\|\, s_0^{(c)},\, *)$, oppure
- $(*,\, s^* \oplus 0^r \,\|\, s_0^{(c)})$.

Nel mondo ideale, la vista di $\mathcal{A}$ è indipendente da $s_0^{(c)}$; di conseguenza, la probabilità di un trascritto cattivo è al più $2q/2^c$.

Per i trascritti *buoni*, valgono le relazioni:

$$p_1(\tau) = 2^{-c} \cdot 2^{-r} \cdot 2^{-n} \cdot p_L, \qquad p_0(\tau) = 2^{-c} \cdot 2^{-r} \cdot q_\tau \cdot p_L,$$

dove p_L rappresenta la probabilità che una permutazione casuale uniforme risulti coerente con le query in L, e q_τ la probabilità che $\mathsf{Sponge}^{\pi_L}_{s_0^{(c)}}(x_1, \ldots, x_\ell) = s^*$.

Poiché τ è buono, l'input $(y^*, s_0^{(c)})$ risulta fresco per π, mentre $s^* \oplus 0^r \,\|\, s_0^{(c)}$ è ancora disponibile; segue che $q_\tau \geq 2^{-n}$. Pertanto, il vantaggio di $\mathcal{A}$ è limitato da:

$$\mathrm{Adv}^{\mathrm{next},\pi}_{\mathrm{Spg}}(\mathcal{A}) \leq \frac{2q}{2^c}.$$

Il risultato mostra che, per stati iniziali casuali, l'uscita prodotta da next è indistinguibile da una stringa uniforme di lunghezza $n + r$, a meno che l'avversario non effettui una query che "colpisca" la capacità interna della permutazione.

Recovering Security – SP

La proprietà di **recovering security** per la costruzione Sponge–PRNG viene dimostrata mediante la tecnica dell'H–coefficient, combinando i risultati di *extraction* e *next security*. La principale differenza risiede nella gestione degli eventi di *hit*, in cui l'avversario interroga l'inversa π^{-1} sul valore finale s^*, che introduce un termine addizionale $\mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A})$ nel bound complessivo.

Come da **Lemma 34** del paper (cfr. Sezione 6.6.4 [7]) si afferma che, per ogni avversario (q, ℓ)–legittimo, vale il seguente vincolo:

$$\mathrm{Adv}^{\mathrm{rec},\pi}_{\mathrm{Spg}}(\mathcal{A}) \leq \delta^{\mathrm{ext}}_{\mathrm{Spg}}(\gamma^*,\, \ell,\, q + \tfrac{r}{n} + 1) + \tfrac{q+\frac{r}{n}+1}{2^n} + 2\,\varepsilon^{\mathrm{next}}_{\mathrm{Spg}}(q + \tfrac{r}{n} + 1),$$

dove $\delta^{\mathrm{ext}}_{\mathrm{Spg}}$ è definito come nel Lemma 29 e $\varepsilon^{\mathrm{next}}_{\mathrm{Spg}}$ come nel Lemma 33.

Dimostrazione. Come nel caso del Lemma 20 (v. Sezione 2.6.4), per $b \in \{0,1\}$ si denoti con H_b l'esperimento di recovering condizionato sul bit segreto b. Si definisce inoltre un esperimento ibrido $H_{1/2}$, in cui l'oracolo di sfida restituisce $\mathsf{next}^{\pi}(U_n)$ all'avversario.

La distanza tra H_0 e $H_{1/2}$ è limitata da una riduzione alla *extraction security*, come nel Lemma 20. In particolare:

$$\varepsilon^{\mathrm{ext}}_{\mathrm{Spg}}(\gamma^*, \ell, q + \tfrac{r}{n} + 1) \leq \delta^{\mathrm{ext}}_{\mathrm{Spg}}(\gamma^*, \ell, q + \tfrac{r}{n} + 1) + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A}_{\mathrm{ext}}).$$

Per limitare la probabilità di un evento di *hit* nel mondo ideale dell'extraction game, si considera un ibrido analogo a H_1. Per la *next security* e per il fatto che un *hit* si verifica con probabilità al più $(q + \frac{r}{n} + 1) \cdot 2^{-n}$, si ottiene:

$$\mathrm{Adv}^{\mathrm{inv},\pi}_{\mathrm{Spg}}(\mathcal{A}_{\mathrm{ext}}) \leq \tfrac{q+r/n+1}{2^n} + \varepsilon^{\mathrm{next}}_{\mathrm{Spg}}(q + r/n + 1).$$

Con un'argomentazione analoga a quella del Lemma 20, si mostra infine che la distanza tra $H_{1/2}$ e H_1 è limitata da un ulteriore termine di *next security*:

$$\mathrm{Adv}(H_{1/2}, H_1) \leq \varepsilon^{\mathrm{next}}_{\mathrm{Spg}}(q + r/n + 1).$$

Applicando la disuguaglianza triangolare si ottiene il bound complessivo sopra riportato.

Preserving Security – SP

La proprietà di **preserving security** per la costruzione Sponge–PRNG si dimostra in modo analogo al caso precedente (*recovering security*), con la sola differenza che le riduzioni coinvolgono la *maintaining security* anziché la *extraction security*. Come nelle altre costruzioni, l'obiettivo è mostrare che, partendo da uno stato ad alta entropia, anche in presenza di input avversari lo stato interno rimanga indistinguibile da uno uniforme e che l'uscita del generatore mantenga la stessa proprietà.

Secondo il **Lemma 35** del paper (cfr. Sezione 6.6.5 [7]) si afferma che, per ogni avversario (q, ℓ)–legittimo, vale:

$$\mathrm{Adv}^{\mathrm{pre},\pi}_{\mathrm{Spg}}(\mathcal{A}) \leq \delta^{\mathrm{mnt}}_{\mathrm{Spg}}(\ell, q + \tfrac{r}{n} + 1) + \tfrac{q+\frac{r}{n}+1}{2^n} + 2 \cdot \varepsilon^{\mathrm{next}}_{\mathrm{Spg}}(q + \tfrac{r}{n} + 1).$$

Dimostrazione. La prova segue passo per passo quella del Lemma 34, sostituendo la riduzione alla *extraction security* con una riduzione alla *maintaining security*. Il primo termine del bound deriva quindi dalla probabilità che l'avversario distingua lo stato interno da uno uniforme dopo l'assorbimento di nuovi input, mentre il termine additivo $\frac{q+\frac{r}{n}+1}{2^n}$ cattura la probabilità di eventi di *hit* nel modello ideale. Infine, i due termini di $\varepsilon^{\mathrm{next}}_{\mathrm{Spg}}$ rappresentano i contributi dovuti alla funzione next nei due esperimenti ibridi.

Combinando tali componenti e applicando la disuguaglianza triangolare, si ottiene il bound complessivo sopra riportato.

2.7 Prove di sicurezza per le Costruzioni IT–Sicure

In questa sezione vengono analizzate le costruzioni di generatori seedless che offrono **sicurezza informazionale**, ossia garanzie di robustezza valide anche contro avversari con capacità di calcolo illimitate. L'obiettivo è illustrare in modo unificato i principi e le tecniche che permettono di dimostrare la sicurezza di tali costruzioni, evidenziando il ruolo giocato dalle grandezze probabilistiche che ne descrivono il comportamento.

Il nucleo tecnico delle dimostrazioni si fonda su tre risultati di base, che mettono in relazione la **distanza statistica** tra distribuzioni con la loro **probabilità di collisione**. Queste relazioni consentono di tradurre il problema dell'indistinguibilità tra due esperimenti in un'analisi puramente probabilistica, misurando quanto una distribuzione reale si discosti da quella perfettamente uniforme.

Nota metodologica. Questa sezione va letta come una sintesi delle dimostrazioni originali presentate in Dodis et al. [7]. Le prove sono qui riportate in forma compatta, mantenendo invariati i risultati e la logica argomentativa di riferimento, ma omettendo i passaggi tecnici e combinatori di dettaglio. L'obiettivo è offrire una visione d'insieme che ne costituisce il nucleo, piuttosto che riprodurre integralmente le dimostrazioni formali.

2.7.0 Preliminari IT

Nella Sezione 2.3.0 sono stati introdotti i concetti di **distanza statistica** e **vantaggio dell'avversario**, che costituiscono la base per l'analisi formale della sicurezza dei modelli seedless. In questa sezione introduciamo un ulteriore strumento matematico, la **collision probability**, che consente di quantificare in modo più diretto la deviazione di una distribuzione casuale rispetto a quella perfettamente uniforme. Attraverso tale misura, è possibile esprimere in forma compatta alcuni risultati classici che verranno poi impiegati nelle prove di robustezza delle costruzioni informazionalmente sicure.

Collision Probability. La probabilità di collisione di una variabile aleatoria X è definita come:

$$\mathrm{Coll}(X) := \sum_x \Pr[X = x]^2.$$

Essa rappresenta la probabilità che due campioni indipendenti estratti secondo la distribuzione di X assumano lo stesso valore. In modo analogo, se X dipende da un'altra variabile aleatoria Y, definiamo la probabilità di collisione condizionata:

$$\mathrm{Coll}(X \mid y) := \sum_x \Pr[X = x \mid Y = y]^2, \qquad \mathrm{Coll}(X \mid Y) := \mathbb{E}_{y \leftarrow Y}[\mathrm{Coll}(X \mid y)].$$

Quest'ultima misura quantifica la concentrazione media di X una volta fissato il valore di Y, e sarà uno strumento fondamentale per analizzare la sicurezza delle costruzioni IT.

Nel seguito riportiamo tre risultati fondamentali che mettono in relazione la distanza statistica e la probabilità di collisione. Essi forniscono una stima superiore della distanza tra due distribuzioni in funzione della loro collision probability, costituendo il punto di partenza per le dimostrazioni successive.

Proposizione 1 (Primo risultato)**.** *Per ogni variabile aleatoria X con dominio di dimensione N, e per U uniformemente distribuita sullo stesso dominio, vale:*

$$\mathrm{SD}(X, U) \leq \frac{1}{2}\sqrt{N \cdot \mathrm{Coll}(X) - 1}.$$

Proposizione 2 (Secondo risultato)**.** *Sia F una variabile aleatoria uniforme su un insieme $\mathcal{F}$. Per ogni variabile X con range di dimensione N (arbitrariamente correlata con F) e per una variabile U uniforme indipendente da F, vale:*

$$\mathrm{SD}((X, F), (U, F)) \leq \frac{1}{2}\sqrt{N \cdot \mathrm{Coll}(X \mid F) - 1}.$$

Proposizione 3 (Terzo risultato)**.** *Siano X e Y due variabili aleatorie con lo stesso dominio, e siano $\mathcal{E}$ e $\mathcal{E}'$ due eventi sui rispettivi spazi di probabilità tali che $\Pr[\mathcal{E}] = \Pr[\mathcal{E}']$. Allora:*

$$\mathrm{SD}(X, Y) \leq \mathrm{SD}(X \mid \mathcal{E},\, Y \mid \mathcal{E}') + \Pr[\overline{\mathcal{E}}].$$

2.7.1 Estrattore Monolitico

Come introdotto nella Sezione 2.3.2, l'estrattore monolitico mono opera con un oracolo casuale

$$G : \{0,1\}^m \to \{0,1\}^n,$$

e costituisce il più semplice esempio di estrattore seedless. In questa sezione ne analizziamo la sicurezza nel modello *informazionalmente sicuro* (IT), mostrando che mono realizza un estrattore $(\gamma^*, q, \varepsilon)$–IT con margine

$$\varepsilon \;\leq\; \frac{1}{2}\sqrt{\frac{2^{-(\gamma^* - n)}}{1-\rho}} \;+\; \rho, \qquad \rho := \frac{q}{2^{\gamma^*}}.$$

Struttura della dimostrazione. La dimostrazione procede limitando la distanza statistica tra l'uscita reale $G(X)$ e una stringa uniforme, condizionata sulla vista dell'avversario. Il punto centrale consiste nell'analizzare la *collision probability* dell'uscita,

$$\mathrm{Coll}(Y) := \Pr[\, Y = Y' \,],$$

dove Y e Y' sono le uscite prodotte da due esecuzioni indipendenti della costruzione. Come mostrato nella **Proposizione 1**, un limite sulla collision probability fornisce direttamente un limite sulla distanza statistica dall'uniforme; l'intera dimostrazione si riduce quindi a stimare $\mathrm{Coll}(Y)$.

Poiché l'avversario può effettuare interrogazioni illimitate all'oracolo G dopo la fase di pre–processing, possiamo assumere che esso abbia accesso all'intera tabella dei valori di G. Il suo vantaggio è quindi limitato dalla distanza statistica tra

$$(\Sigma,\, Y_0,\, \mathcal{L}_1,\, G) \qquad \text{e} \qquad (\Sigma,\, Y_1,\, \mathcal{L}_1,\, G),$$

dove Σ è lo stato interno dell'avversario, $\mathcal{L}_1$ la lista delle query effettuate prima della sfida e $Y_0 = G(X)$, mentre Y_1 è una stringa uniforme.

Il primo passo consiste nell'isolare l'evento

$$\mathcal{E} := \text{“l'avversario non interroga l'input } X\text{”}.$$

La probabilità che $\mathcal{E}$ fallisca è controllata dalla legittimità dell'avversario:

$$\Pr[\overline{\mathcal{E}}] \;\le\; q \cdot \mathrm{Pred}(X \mid \Sigma, \mathcal{L}_1) \;\le\; q \cdot 2^{-\gamma^*} = \rho.$$

Applicando la decomposizione per la distanza statistica (v. **Proposizione 3**), si riduce il problema a stimare

$$\mathrm{SD}\Big((Y_0, G) \mid \mathcal{E},\ (Y_1, G) \mid \mathcal{E}\Big).$$

Fissati lo stato avversario Σ e la lista delle query $\mathcal{L}_1$, la tabella dei valori di G condizionata a tali query è uniforme tra tutte le funzioni compatibili con $\mathcal{L}_1$. In queste condizioni il valore $Y_0 = G(X)$ può essere analizzato tramite la collision probability condizionata: applicando la **Proposizione 2**, si ottiene

$$\mathrm{SD}\big((Y_0, G) \mid z\mathcal{E},\ (Y_1, G) \mid z\mathcal{E}\big) \le \frac{1}{2}\sqrt{2^n \cdot \mathrm{Coll}(Y_0 \mid z, \mathcal{E}) - 1}.$$

Stimando la collision probability tramite la predicabilità dell'input, si ottiene che essa coincide con $p_z = \mathrm{Pred}(X \mid Z = z, \mathcal{E})$. La **disuguaglianza di Jensen**[11] permette di passare alla media rispetto a z, ottenendo

$$\mathrm{SD}\big((Y_0, G) \mid z\mathcal{E},\ (Y_1, G) \mid z\mathcal{E}\big) \;\le\; \frac{1}{2}\sqrt{2^n \cdot \mathrm{Pred}(X \mid Z\mathcal{E})}.$$

Dalla γ^*–legittimità segue $\mathrm{Pred}(X \mid Z) \le 2^{-\gamma^*}$, mentre il fattore $(1-\rho)^{-1}$ deriva dal condizionamento sull'evento $\mathcal{E}$. Combinando questo contributo con $\Pr[\overline{\mathcal{E}}] \le \rho$ si ottiene il margine di sicurezza riportato in apertura della sezione.

2.7.2 Nozioni Intermedie di IT-Sicurezza per PRNG

Analogamente al caso computazionale (v. Sezione 2.6.3), è utile introdurre una nozione intermedia di sicurezza anche per i PRNG informazionalmente sicuri (IT), denominata **recovering security**. Nel caso computazionale, tale proprietà richiede che l'uscita della funzione next appaia casuale una volta che sufficiente entropia sia stata accumulata nello stato del PRNG. Nel contesto IT, la condizione è leggermente più "debole": si richiede soltanto che l'uscita sia indistinguibile da una distribuzione del tipo $(0^n, U_r)$, dove U_r è una stringa uniforme di lunghezza r.

In altre parole, ogni chiamata a next azzera lo stato interno (e quindi l'entropia accumulata), senza perdita di generalità, poiché la definizione di legittimità considera ogni invocazione **indipendente**.

[11] La disuguaglianza di Jensen afferma che, per ogni funzione concava f e ogni variabile aleatoria non negativa U, vale $\mathbb{E}[f(U)] \le f(\mathbb{E}[U])$. Nel nostro caso si applica alla funzione $f(u) = \sqrt{u}$ per passare dal valore $\sqrt{p_z}$ alla media rispetto alla distribuzione di Z, ottenendo un bound in termini di $\mathrm{Pred}(X \mid Z\mathcal{E})$.

The PRNG Recovering-IT Game	
init	**chall**$(s_0, x_1, \ldots, x_\ell)$
$b \leftarrow \{0,1\}$	**for** $i = 1, \ldots, \ell$ **do**
	$s_i \leftarrow \mathsf{refresh}(s_{i-1}, x_i)$
	if $b = 0$ **then**
	return $\mathsf{next}(s_\ell)$
	else
	$y \leftarrow \{0,1\}^r$
	return $(0^n, y)$

Tabella 2.4. Oracoli del gioco di recupero (*Recovering-IT Game*) per i PRNG seedless.

Il recovering game per PRNG IT-sicuri è rappresentato nella Tabella 2.4. Il vantaggio di un avversario $\mathcal{A}$ in tale gioco è indicato con $\mathrm{Adv}^{\text{rec-IT},P}_{\text{PRNG}}(\mathcal{A})$ ed un avversario $\mathcal{A}$ si dice γ^*-*legittimo* se vale:

$$H_\infty(X_1, \ldots, X_\ell \mid \Sigma \mathcal{L} S_0) \;\geq\; \gamma^*,$$

dove:

- Σ è lo stato interno dell'avversario immediatamente prima della chiamata a chall;
- $\mathcal{L}$ è la lista delle query e risposte che $\mathcal{A}$ ha effettuato verso l'oracolo P prima della chiamata a chall;
- S_0 è lo stato iniziale fornito dall'avversario.

Un avversario (q, ℓ)–IT–legittimo è quindi uno che può effettuare al massimo q query a P prima della sfida (e un numero arbitrario successivamente), e il cui input alla funzione chall è composto da al più ℓ blocchi.

Definizione 2.11 (Recovering security IT)**.** Una costruzione di $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ è detta $(\gamma^*, q, \ell, \varepsilon)$–**IT–recovering** nel modello P se, per ogni avversario γ^*–legittimo, vale:

$$\mathrm{Adv}^{\text{rec-IT},P}_{\text{PRNG}}(\mathcal{A}) \;\leq\; \varepsilon.$$

Si osservi che la recovering security per i PRNG informazionalmente sicuri è, a parte per differenze sintattiche, equivalente alla sicurezza dell'estrattore associato al PRNG stesso. Infatti, sostituendo l'algoritmo next con una procedura finalize che ignora lo stato interno, si ottiene una funzione estrattrice il cui comportamento risulta indistinguibile da quello di un PRNG IT-sicuro.

Teorema di Riduzione da Recovering a Robustezza IT. Sia $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ un generatore tale che $\mathsf{refresh}$ effettui α chiamate all'oracolo P e next ne effettui β. Se la costruzione è $(\gamma^*, q, \ell, \varepsilon_{\text{rec}})$–IT–recovering nel modello P, allora essa è anche $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$–IT–robusta, con:

$$\varepsilon_{\text{rob}} \;\leq\; t \cdot \varepsilon_{\text{rec}}.$$

Il risultato mostra che la robustezza IT di un PRNG può essere ricondotta alla sua proprietà di recovering security, tramite un'analisi di tipo unione.

2.7.3 IT–PRNGs basati su Merkle-Damgård

In questa sezione analizziamo, in maniera generale (cfr. Sezione 7.4 [7]), la robustezza informazionale della costruzione basata sul paradigma di **Merkle–Damgård**, già presentata nella Sezione 2.5.1. Si ricordi che la costruzione $\text{MD}_r = (\mathsf{refresh}, \mathsf{next})$ utilizza una funzione di compressione modellata tramite un oracolo casuale $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, alla quale l'avversario ha accesso illimitato. La sua definizione formale è riportata in Costruzione 5.

Ricordiamo inoltre, rispetto al caso computazionale (cfr. Sezione 2.6.4), la procedura $\mathsf{refresh}$ rimane invariata, mentre la funzione next viene modificata per ottenere garanzie di sicurezza anche contro avversari non limitati nel calcolo (si limita a *troncare* lo stato ai primi r bit e ad azzerare l'intero stato interno).

In questo modo, l'avversario non può più "seguire" l'evoluzione dello stato interno interrogando l'oracolo F, poiché l'output osservabile è solo una porzione dello stato e, dopo ogni chiamata a next, lo stato viene resettato a 0^n.

L'obiettivo della sezione corrispondente del paper è mostrare che MD_r realizza un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob-it}})$–robusto nel modello ideale F, con margine di sicurezza:

$$\varepsilon_{\text{rob-it}} \leq \frac{t}{2}\sqrt{\frac{2^{r-\gamma^*}}{1-\rho} + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^{2n}} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 \cdot \frac{\tilde{q}^2 2^r}{2^{2n}}} + t\rho,$$

$$\textit{per } \rho = \frac{\tilde{q}^2}{2^r}, \qquad \textit{dove } \tilde{q} = q + t\ell$$

Struttura della dimostrazione. La dimostrazione segue lo schema informazionale delineato nei preliminari (cfr. Sezione 2.7.0). L'obiettivo è limitare la distanza statistica tra l'uscita reale del PRNG e una stringa uniforme, riconducendo tale quantità alla predicibilità dell'input accumulato e alla probabilità di collisione dell'uscita troncata. Per descrivere formalmente il gioco di recovering si considerano le seguenti variabili casuali. La funzione di compressione F è l'oracolo casuale a disposizione dell'avversario; ℓ è la lunghezza, variabile ma limitata, della sequenza di input assorbita dal PRNG; $\bar{X} = (\bar{X}_1, \ldots, \bar{X}_\ell)$ rappresenta gli input stessi;

$\tilde{Y}_\ell$ è l'uscita troncata ottenuta applicando next allo stato finale della catena Merkle–Damgård; $Z = (\Sigma, L, S_0)$ codifica la vista dell'avversario prima della sfida (stato interno, lista di query e stato iniziale); infine U_r è una stringa uniforme di r bit che rappresenta l'uscita ideale. Il vantaggio dell'avversario è dato dalla distanza statistica tra

$$(\tilde{Y}_\ell, Z, F) \qquad \text{e} \qquad (U_r, Z, F),$$

e la dimostrazione procede decomponendo tale distanza in contributi più gestibili.

Evento favorevole ed eliminazione dei casi rari. Il primo passo consiste nell'isolare un evento "favorevole" che escluda le collisioni indesiderate tra le risposte dell'oracolo F osservate dall'avversario. Si definisce $\mathcal{E}$ come "*tutte le risposte alle query di F hanno primi r bit distinti*".

Poiché l'avversario effettua al più q interrogazioni prima della sfida, la probabilità che due output troncati coincidano è limitata da

$$\Pr[\overline{\mathcal{E}}] \le \frac{q^2}{2^r}, \qquad \rho := \frac{q^2}{2^r}.$$

Applicando la Proposizione 3, la distanza statistica si scompone come:

$$\mathrm{SD}\big((\tilde{Y}_\ell, Z, F), (U_r, Z, F)\big) \le \mathrm{SD}\big((\tilde{Y}_\ell, Z, F) \mid \mathcal{E},\, (U_r, Z, F) \mid \mathcal{E}\big) + \rho.$$

La parte principale della dimostrazione può dunque concentrarsi sul caso in cui $\mathcal{E}$ si verifica.

Per analizzare la distanza condizionata, si fissa una vista avversaria specifica $Z = z$. La distanza statistica rimanente può essere espressa come una media rispetto a tutte le viste compatibili con $\mathcal{E}$:

$$\mathrm{SD}\big((\tilde{Y}_\ell, Z, F) \mid \mathcal{E},\, (U_r, Z, F) \mid \mathcal{E}\big) = \sum_{z \in \mathcal{E}} \Pr[Z = z \mid \mathcal{E}] \cdot \mathrm{SD}\big((\tilde{Y}_\ell, F) \mid z,\, (U_r, F) \mid z\big).$$

Diventa quindi sufficiente limitare $\mathrm{SD}((\tilde{Y}_\ell, F) \mid z, (U_r, F) \mid z)$ per ogni vista fissata.

Il parametro cruciale in questa fase è

$$p_z := \mathrm{Pred}(\bar{X} \mid Z = z),$$

che misura quanto l'avversario possa predire l'input dopo aver osservato z. La definizione di avversario γ^*–legittimo garantisce che $\mathbb{E}[p_Z] \le 2^{-\gamma^*}$, e dunque la predicibilità media è esponenzialmente piccola nel parametro di sicurezza.

A questo punto si collega la distanza statistica alla collision probability dell'uscita troncata. Condizionando su $Z = z$ e sull'evento $\mathcal{E}$, la funzione F è distribuita uniformemente tra tutte quelle compatibili con le query osservate. La Proposizione 2 consente quindi di limitare la distanza statistica in termini della collision probability condizionata $\mathrm{Coll}(\tilde{Y}_\ell \mid F, z)$. Il **Lemma 38** mostra che, per ogni $z \in \mathcal{E}$,

$$\mathrm{SD}\big((\tilde{Y}_\ell, F) \mid z,\, (U_r, F) \mid z\big) \le \frac{1}{2}\sqrt{2^r p_z \;+\; \ell \cdot d'(\ell)\,\frac{2^r}{2^n} \;+\; 64\ell^4\,\frac{2^r}{2^{2n}} \;+\; 16\ell^2 q^2\,\frac{2^r}{2^{2n}}}.$$

I termini che coinvolgono $d'(\ell)$, ℓ^4 e $\ell^2 q^2$ rappresentano collisioni "strutturali" all'interno della computazione Merkle–Damgård e verranno analizzati tramite i grafi di struttura nella parte successiva. Infine, si media rispetto alla distribuzione di $Z \mid \mathcal{E}$: poiché la radice quadrata è concava, la disuguaglianza di Jensen permette di sostituire $\mathbb{E}[\sqrt{2^r p_Z}]$ con $\sqrt{2^{r-\gamma^*}}$. Unendo tale contributo con la probabilità dell'evento complementare $\overline{\mathcal{E}}$, pari a $\rho = q^2/2^r$, si ottiene esattamente il margine di recovering security enunciato nel **Lemma 37**, secondo cui ogni avversario γ^*–legittimo (q, ℓ) ottiene un vantaggio estremamente contenuto, determinato dalla predicibilità residua dell'input e corretto solo da termini di ordine minore.

Grafi di struttura e accidents. Per analizzare in modo fine la probabilità di collisione fra due computazioni indipendenti della catena Merkle–Damgård, il paper introduce un modello combinatorio basato sui *grafi di struttura*. Dato un oracolo F compatibile con la lista di query avversarie L e due sequenze di input $x = (x_1, \ldots, x_\ell)$ e $x' = (x'_1, \ldots, x'_{\ell'})$, si definisce un grafo orientato $G_F(x, x')$ i cui nodi rappresentano stati n-bit e i cui archi descrivono le transizioni $s_i = F(s_{i-1}, x_i)$ e $s'_j = F(s'_{j-1}, x'_j)$. Gli archi vengono distinti in:

- *blue edges*, che corrispondono alle valutazioni di F già presenti nella lista di query avversarie, e
- *red edges*, che rappresentano valutazioni "fresche" della funzione di compressione generate durante l'esecuzione di Merkle–Damgård.

Nel grafo così costruito compaiono due percorsi distinti, uno per la computazione su x e uno per quella su x'. Un concetto cruciale è quello di *accident*: un accident si verifica quando un red edge punta a un nodo già presente nel grafo, cioè quando una valutazione fresca di F collide con una struttura già formata. Si indica con $\mathrm{Acc}(G_F)$ il numero totale di accidents. Poiché i due percorsi possono fondersi solo tramite tali eventi, il controllo degli accidents è essenziale per stimare la probabilità di collisione dell'output.

A questo punto, la dimostrazione analizza una serie di lemmi che consentono di decomporre e quantificare la probabilità che il grafo $G_F(x, x')$ risulti effettivamente collidente. Il primo risultato, il **Lemma 39**, separa il contributo dovuto alla struttura del grafo da quello dovuto al troncamento finale:

$$\Pr[\tilde{Y}_\ell = \tilde{Y}'_\ell] \;\le\; \Pr\big[G_F(\bar{x}, \bar{x}') \in \mathrm{Coll}_L(\bar{x}, \bar{x}')\big] \;+\; \frac{1}{2^r}.$$

La parte "strutturale" della collisione è dunque interamente contenuta nella probabilità che i due percorsi terminino nello stesso nodo del grafo di struttura; il termine 2^{-r} riflette la possibilità di coincidenze casuali dopo il troncamento ai primi r bit.

Per valutare il primo termine, il **Lemma 40** collega la probabilità del grafo al numero dei suoi accidents: se un grafo H presenta $\mathrm{Acc}(H) = a$, allora

$$\Pr_F[G_F(\bar{x}, \bar{x}') = H] = \frac{1}{2^{n \cdot Acc(H)}}.$$

Ogni accident comporta infatti che una valutazione fresca di F assuma un valore specifico fra 2^n possibilità; grafi con due o più accidents risultano quindi esponenzialmente meno probabili.

Su questa base, il **Lemma 41** fornisce un bound complessivo sulla probabilità di osservare almeno due accidents:

$$\Pr_F[\mathrm{Acc}(G_F) \ge 2] \;\le\; \frac{64\ell^4}{2^{2n}} \;+\; \frac{16\ell^2 q^2}{2^{2n}}.$$

Il primo termine riflette il numero di coppie di posizioni in cui potrebbero verificarsi due accidents lungo i percorsi MD; il secondo ingloba le possibili interazioni con le q query avversarie. Si tratta in ogni caso di eventi estremamente rari, per cui la parte dominante dell'analisi riguarda i grafi collidenti con un solo accident.

È proprio questo il caso studiato dal **Lemma 42**, che introduce la distinzione fra tutti i blue edges del grafo, $\mathcal{E}_B$, e quelli *iniziali*, $\mathcal{E}'_B$, cioè gli archi blu che compaiono prima del primo red edge. Il lemma mostra che, se un grafo è collidente e ha esattamente un accident, allora necessariamente $\mathcal{E}'_B = \mathcal{E}_B$.

Non possono quindi esistere blue edges "interni" (posti dopo il red edge responsabile dell'unico accident), poiché la loro presenza forzerebbe la comparsa di un secondo accident, contraddicendo l'ipotesi. Questo vincolo riduce drasticamente le configurazioni possibili e consente di limitare la parte più delicata del conteggio combinatorio alle strutture in cui la complessità è concentrata nella porzione di grafo precedente all'accident.

Combinando i lemmi precedenti con un ulteriore conteggio sulle posizioni ammissibili dell'unico accident e sulle strutture indotte dalle possibili periodicità dei percorsi, si ottiene il bound

$$\Pr[\tilde{Y}_\ell = \tilde{Y}'_\ell] \;\le\; \frac{1}{2^r} \;+\; \frac{\ell \cdot d'(\ell)}{2^n} \;+\; \frac{64\ell^4}{2^{2n}} \;+\; \frac{16\ell^2 q^2}{2^{2n}}.$$

I termini che coinvolgono $d'(\ell)$, ℓ^4 e $\ell^2 q^2$ quantificano la probabilità che si verifichino collisioni "strutturali" nella computazione Merkle–Damgård. Questo risultato fornisce esattamente la stima su $\Pr[\tilde{Y}_\ell = \tilde{Y}'_\ell]$ utilizzata nel Lemma 38 per ottenere il bound sulla distanza statistica.

2.7.4 IT–PRNGs basati su Merkle–Damgård con Davies–Meyer

In questa sezione analizziamo, in maniera generale (cfr. Sezione 7.5 [7]), la robustezza informazionale della costruzione $\mathrm{DM}_r = (\mathsf{refresh}, \mathsf{next})$, che instanzia Merkle–Damgård utilizzando come funzione di compressione la trasformazione di **Davies–Meyer** $F(x, y) = E(x, y) \oplus y$, dove E è un cifrario ideale $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. La definizione formale della costruzione è riportata in Costruzione 5. Anche in questo caso, la procedura $\mathsf{refresh}$ rimane invariata, mentre la funzione next restituisce gli r bit più significativi dello stato e azzera l'intero registro interno.

La dimostrazione segue da vicino quella del caso Merkle–Damgård classico (Sezione 2.7.3): le tecniche di decomposizione della distanza statistica, l'uso della collision probability e l'applicazione delle Proposizioni 2 e 3 rimangono identiche. L'unica differenza sostanziale riguarda **l'analisi combinatoria dei grafi di struttura**, che deve essere adattata alla natura del cifrario ideale.

L'obiettivo è mostrare che la costruzione DM_r realizza un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob-it}})$–robusto nel modello ideale E. Il margine di sicurezza ottenuto nella sezione corrispondente del paper è

$$\varepsilon_{\text{rob-it}} \le \frac{t}{2}\sqrt{\frac{2^{\,r-\gamma^*}}{1-\rho} \;+\; \ell \cdot d'(\ell)\frac{2^r}{2^{\,2n-1}} \;+\; 64\ell^4\frac{2^r}{2^{\,2n-2}} \;+\; 16\ell^2\frac{\tilde{q}^{\,2}2^r}{2^{\,2n-2}}} \;+\; t\rho\;,$$

$$\textit{dove } \rho = \frac{\tilde{q}^{\,2}}{2^r}, \qquad \textit{con } \tilde{q} = q + t\ell.$$

Struttura della dimostrazione. Come nel caso precedente, si considera la distanza statistica tra l'uscita reale troncata e una stringa uniforme,

$$(\tilde{Y}_\ell, Z, E) \qquad \text{e} \qquad (U_r, Z, E),$$

dove $X = (X_1, \ldots, X_\ell)$ è la sequenza di input assorbiti, $\tilde{Y}_\ell$ l'output troncato prodotto da next, $Z = (\Sigma, L, S_0)$ la vista dell'avversario prima della sfida, ed E un cifrario ideale compatibile con la lista di query L. La dimostrazione procede esattamente come nel caso MD: si isola un evento favorevole, si condiziona sulla vista avversaria e si collega la distanza statistica alla collision probability dell'uscita troncata.

Evento favorevole ed eliminazione dei casi rari. Si definisce l'evento $\mathcal{E}$ come "*tutte le risposte alle query avversarie hanno primi r bit distinti*". Poiché l'avversario esegue al più q interrogazioni prima della sfida, la probabilità di una collisione sugli r bit più significativi è

$$\Pr[\overline{\mathcal{E}}] \le \frac{q^2}{2^r}, \qquad \rho := \frac{q^2}{2^r}.$$

La Proposizione 3 permette di decomporre la distanza statistica come

$$\mathrm{SD}\big((\tilde{Y}_\ell, Z, E), (U_r, Z, E)\big) \le \mathrm{SD}\big((\tilde{Y}_\ell, Z, E) \mid \mathcal{E},\ (U_r, Z, E) \mid \mathcal{E}\big) + \rho.$$

Fissata una vista avversaria $Z = z$, la distanza condizionata si riscrive come media sulle viste compatibili:

$$\mathrm{SD}\big((\tilde{Y}_\ell, Z, E) \mid \mathcal{E},\ (U_r, Z, E) \mid \mathcal{E}\big) = \sum_{z \in \mathcal{E}} \Pr[Z = z \mid \mathcal{E}]\, \mathrm{SD}\big((\tilde{Y}_\ell, E) \mid z,\ (U_r, E) \mid z\big).$$

Si introduce quindi come nel caso MD il parametro di predicibilità

$$p_z := \mathrm{Pred}(X \mid Z = z),$$

e la γ^*–legittimità dell'avversario implica $\mathbb{E}[p_Z] \le 2^{-\gamma^*}$.

Condizionando su $Z = z$ e sull'evento favorevole, il cifrario E è distribuito uniformemente tra tutti quelli compatibili con la lista di query L. La Proposizione 2 permette allora di esprimere la distanza statistica in funzione della collision probability dell'uscita troncata.

Il **Lemma 44** mostra che, per ogni $z \in \mathcal{E}$,

$$\mathrm{SD}\big((\tilde{Y}_\ell, E) \mid z,\ (U_r, E) \mid z\big) \le \frac{1}{2}\sqrt{2^r p_z \ +\ \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} \ +\ 64\ell^4\frac{2^r}{2^{2n-2}} \ +\ 16\ell^2 q^2 \frac{2^r}{2^{2n-2}}}.$$

La media rispetto a $Z \mid \mathcal{E}$ e la concavità della radice quadrata (Jensen) permettono di sostituire $\mathbb{E}[\sqrt{2^r p_Z}]$ con $\sqrt{2^{\,r-\gamma^*}}$. Combinando questo contributo con $\Pr[\overline{\mathcal{E}}] = \rho$ si ottiene il margine di recovering security enunciato nel **Lemma 43**.

Grafi di struttura e accidents. Per analizzare la probabilità di collisione fra due computazioni indipendenti della catena Davies–Meyer si utilizza, come nel caso MD, il modello dei grafi di struttura. In questa instanziazione, però, una valutazione fresca del cifrario ideale non distribuisce l'uscita su tutti i 2^n possibili valori, ma su un insieme di cardinalità 2^{n-1}. Questa caratteristica modifica la probabilità degli *accidents* e porta alla variante del Lemma 40 formalizzata nel **Lemma 45**. Si mostra che, se un grafo di struttura H contiene $\mathrm{Acc}(H) = a$ accidents, allora

$$\Pr_E[G_E(\bar{x}, \bar{x}') = H] = \frac{1}{2^{(n-1)\cdot a}}.$$

Inserendo questo bound nella dimostrazione dei Lemmi 41 e 42 (si noti che quest'ultimo rimane invariato), si ottiene il corrispondente limite sulla collision probability:

$$\Pr[\tilde{Y}_\ell = \tilde{Y}'_\ell] \leq \frac{1}{2^r} + \frac{\ell\, d'(\ell)}{2^{\,n-1}} + \frac{64\ell^4}{2^{\,2n-2}} + \frac{16\ell^2 q^2}{2^{\,2n-2}}.$$

Questo è il bound strutturale utilizzato nel Lemma 44 per ottenere il limite finale sulla distanza statistica e completare la prova.

2.7.5 IT–PRNGs basati su Sponge

In questa sezione analizziamo, in maniera generale (cfr. Sezione 7.6 [7]), la robustezza informazionale della costruzione basata sul paradigma **Sponge** nel modello di permutazione ideale. Si ricordi che la costruzione $\mathsf{Spg}_r = (\mathsf{refresh}, \mathsf{next})$ utilizza una permutazione pubblica $\pi : \{0,1\}^n \to \{0,1\}^n$, organizzando lo stato in r bit di *rate* e $c = n - r$ bit di *capacity*.

La definizione formale è riportata in Costruzione 9: la procedura refresh assorbe blocchi di input sommando (via XOR) sui bit di rate prima di applicare π, mentre next restituisce gli r bit di rate e azzera lo stato. Come nei casi MD e MD–DM, la funzione next viene definita in modo da troncare e *resettare* lo stato interno ad ogni uscita, impedendo all'avversario di seguire l'evoluzione della permutazione semplicemente tramite query all'oracolo π.

L'obiettivo della sezione corrispondente del paper è mostrare che Spg_r realizza un PRNG $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob-it}})$–robusto nel modello ideale π, con margine di sicurezza

$$\varepsilon_{\text{rob-it}} \leq \frac{t}{2}\sqrt{\frac{2^{\,r-\gamma^*}}{1-\rho} \;+\; \frac{\ell\cdot(\ell+\tilde{q})}{2^{\,c-1}}} \;+\; t\rho,$$

$$\textit{dove } \rho = \frac{\tilde{q}^{\,2}}{2^c}, \qquad \textit{con } \tilde{q} = q + t\ell.$$

Struttura della dimostrazione. La dimostrazione segue la stessa impostazione informazionale già vista per il caso Merkle–Damgård (cfr. Sezione 2.7.3). Si considera la distanza statistica tra l'uscita reale del PRNG e una stringa uniforme,

$$(\tilde{Y}_\ell, Z, \pi) \qquad \text{e} \qquad (U_r, Z, \pi),$$

dove $\bar{X} = (\bar{X}_1, \ldots, \bar{X}_\ell)$ è la sequenza di blocchi assorbiti, $\tilde{Y}_\ell$ è l'uscita troncata prodotta da next, $Z = (\Sigma, L, S_0)$ rappresenta la vista dell'avversario prima della

sfida (stato interno, lista di query e stato iniziale), ed π è una permutazione ideale compatibile con la lista di query L. Come in precedenza, la prova procede isolando un evento favorevole, condizionando sulla vista avversaria e collegando la distanza statistica alla probabilità di collisione dell'uscita troncata.

Evento favorevole ed eliminazione dei casi rari. Per tenere sotto controllo le collisioni sui bit di *capacity*, il paper introduce un evento "favorevole" che esclude i casi in cui le query dell'avversario si sovrappongono su questa parte dello stato. Si definisce l'evento $\mathcal{E}$:= "*tutte le query in L non collidono sugli ultimi c bit dello stato*".

In termini di Sponge, i nodi di stato sono raggruppati in *supernodi* che condividono gli stessi c bit di capacity: l'evento $\mathcal{E}$ richiede che ogni query dell'avversario atterri in un supernodo diverso.

Poiché l'avversario può effettuare al più q interrogazioni alla permutazione, la probabilità che due query cadano nello stesso supernodo è limitata da

$$\Pr[\overline{\mathcal{E}}] \le \frac{q^2}{2^c}, \qquad \rho := \frac{q^2}{2^c}.$$

Applicando la Proposizione 3, otteniamo la consueta decomposizione:

$$\mathrm{SD}\big((\tilde{Y}_\ell, Z, \pi), (U_r, Z, \pi)\big) \le \mathrm{SD}\big((\tilde{Y}_\ell, Z, \pi) \mid \mathcal{E},\ (U_r, Z, \pi) \mid \mathcal{E}\big) + \rho.$$

La parte principale dell'analisi può dunque concentrarsi sul caso in cui $\mathcal{E}$ si verifica.

Fissata una vista avversaria $Z = z$, la distanza statistica condizionata si riscrive come media sulle viste compatibili:

$$\mathrm{SD}\big((\tilde{Y}_\ell, Z, \pi) \mid \mathcal{E},\ (U_r, Z, \pi) \mid \mathcal{E}\big) = \sum_{z \in \mathcal{E}} \Pr[Z = z \mid \mathcal{E}] \cdot \mathrm{SD}\big((\tilde{Y}_\ell, \pi) \mid z,\ (U_r, \pi) \mid z\big).$$

Come nel caso MD, il parametro cruciale è

$$p_z := \mathrm{Pred}(\bar{X} \mid Z = z),$$

che misura la predicibilità dell'input dopo aver fissato la vista z. La γ^*–legittimità dell'avversario nel modello IT implica $\mathbb{E}[p_Z] \le 2^{-\gamma^*}$ e, più precisamente, che l'entropia minima condizionata $H_\infty(\bar{X} \mid Z\mathcal{E})$ degrada di al più $\log(1-\rho)^{-1}$ rispetto a γ^*.

Condizionando su $Z = z$ e sull'evento $\mathcal{E}$, la permutazione π è distribuita uniformemente tra tutte le permutazioni compatibili con la lista di query L. In questa situazione si può applicare la Proposizione 2, che esprime la distanza statistica in funzione della collision probability condizionata dell'uscita troncata. Il **Lemma 47** mostra che, per ogni $z \in \mathcal{E}$,

$$\mathrm{SD}\big((\tilde{Y}_\ell, \pi) \mid z,\ (U_r, \pi) \mid z\big) \le \frac{1}{2}\sqrt{2^r p_z \ +\ \frac{\ell \cdot (\ell + q)}{2^{\,c-1}}}.$$

Il termine $2^r p_z$ riflette direttamente la predicibilità dell'input, mentre $\ell \cdot (\ell + q)/2^{c-1}$ cattura la probabilità di collisioni strutturali tra due valutazioni della Sponge, anche tenendo conto delle q query avversarie.

Mediando rispetto a $Z \mid \mathcal{E}$ e usando la concavità della radice quadrata (disuguaglianza di Jensen), si può sostituire $\mathbb{E}[\sqrt{2^r p_Z}]$ con una quantità controllata in termini di $2^{\,r-\gamma^*}/(1-\rho)$. Combinando questo contributo con la probabilità dell'evento complementare $\overline{\mathcal{E}}$, pari a $\rho = q^2/2^c$, si ottiene il margine di recovering security enunciato nel **Lemma 46**: per ogni avversario γ^*–legittimo (q, ℓ), il vantaggio nel gioco di recovering IT è limitato da una quantità dominata dal termine principale $\sqrt{2^{\,r-\gamma^*}/(1-\rho)}$, corretta solo da contributi proporzionali a $\ell \cdot (\ell+q)/2^{c-1}$ e alla probabilità ρ dei casi rari.

Analisi delle collisioni via supernodi. Il bound su $\ell \cdot (\ell+q)/2^{c-1}$ deriva da un'analisi più fine delle collisioni fra due valutazioni indipendenti della Sponge sugli input x e x'. Per questo, il paper introduce il concetto di *supernodo*: gli n bit di stato sono divisi in r bit di rate e c bit di capacity, e tutti i nodi che condividono gli stessi c bit finali vengono raggruppati nello stesso supernodo.

Condizionando su $Z = z$ e su una permutazione π compatibile con L, si considerano due valutazioni della Sponge sugli input x e x', entrambe troncate ai primi r bit. Il **Lemma 48** mostra che la probabilità che le due computazioni terminino nello stesso supernodo e producano lo stesso output troncato è limitata da

$$\Pr[\tilde{Y}_\ell = \tilde{Y}'_\ell] \;\le\; \frac{\ell\,(\ell+q)}{2^{\,c-1}} \;+\; \frac{1}{2^r}.$$

Il termine $1/2^r$ corrisponde alla possibilità che i due output coincidano semplicemente per effetto del troncamento, mentre $\ell(\ell+q)/2^{c-1}$ nasce dal conteggio dei modi in cui le sequenze di stati possono "incontrarsi" nei supernodi di capacity, tenendo conto sia delle valutazioni interne sia delle query in L.

Inserendo questo bound strutturale nella Proposizione 2 e nella catena di disuguaglianze precedente, si ottiene il Lemma 47 e, da questo, il Lemma 46 sulla recovering security per una singola istanza Sponge. Infine, applicando il teorema di riduzione generale per la robustezza (analogo al caso Merkle–Damgård, con la sostituzione $q \leftarrow \tilde{q} = q + t\ell$ e $\rho \leftarrow \tilde{q}^{\,2}/2^c$), si ricava il margine di sicurezza globale riportato all'inizio della sezione per la costruzione Spg_r.

Capitolo 3

Critiche e limiti dei Seedless PRNG

Se da un lato i generatori pseudocasuali *seedless* mostrano caratteristiche innovative rispetto ai modelli classici *seeded*, dall'altro non sono mancati tentativi di evidenziare limiti e criticità di questo paradigma. In questo capitolo verranno analizzati i principali risultati della letteratura più recente, con particolare attenzione agli attacchi di tipo *Premature-Next*, ai limiti teorici messi in luce dagli studi di Coretti et al. [5], e alle implicazioni di tali considerazioni per l'utilizzo pratico di questi generatori.

3.1 Il problema *Premature-Next* e perché conta

Come discusso nel Capitolo 2, le costruzioni *seedless* garantiscono sicurezza finché l'accumulo di entropia procede regolarmente e l'algoritmo `next` viene invocato soltanto quando lo stato interno contiene una quantità sufficiente di entropia. Tuttavia, questa ipotesi di comportamento ideale non rispecchia sempre la realtà operativa dei sistemi moderni.

Gli attacchi di tipo *Premature-Next*, introdotti da Kelsey, Schneier, Wagner e Hall [15], si verificano quando un generatore viene sollecitato a produrre output prima di aver accumulato abbastanza entropia per garantire la sicurezza. Questo scenario è tipico, ad esempio, nelle fasi di avvio di un sistema operativo, quando le fonti entropiche disponibili (interruzioni, movimenti del mouse, tempi di I/O, ecc.) sono ancora scarse o altamente prevedibili. In tali condizioni, l'output generato può risultare insufficiente dal punto di vista della casualità e, nei casi peggiori, può fornire all'avversario informazioni utili per ricostruire lo stato interno del generatore.

Le implementazioni pratiche hanno adottato diverse strategie per mitigare il problema. Il dispositivo `/dev/random` di Linux, ad esempio, blocca l'erogazione di output fino al raggiungimento di una soglia minima di entropia, evitando che il generatore produca bit in condizioni di scarsa imprevedibilità. Tuttavia, questo approccio non offre una garanzia teorica: stimare con esattezza la quantità di entropia accumulata in un sistema è, di fatto, un problema impossibile da risolvere in modo affidabile [9], e diversi studi hanno mostrato che tale stima può essere facilmente ingannata.

Un tentativo più sofisticato è quello dei generatori Yarrow e Fortuna [12, 31], che distribuiscono l'entropia in più *pool* indipendenti e aggiornano un registro principale in modo periodico. Ogni evento entropico viene assegnato a un pool secondo uno schema *round-robin*; periodicamente, il contenuto di alcuni pool viene usato per aggiornare il registro. In questo modo, anche se una chiamata prematura compromette il pool appena "svuotato", l'entropia presente nei pool non ancora impiegati rimane protetta, introducendo un principio di *separazione temporale*.

Questo approccio, formalizzato da Dodis et al. [11], costituisce la base del modello di robustezza successivamente esteso, come vedremo nella prossima sezione. In tale impostazione, gli input entropici sono distribuiti su più *pool* che, con diverse frequenze, aggiornano un registro centrale dal quale vengono estratti i bit pseudocasuali. Yarrow e Fortuna utilizzano strategie di *scheduling* deterministiche per assegnare ogni input a un pool e decidere quando ciascun pool contribuisce al registro, mentre la robustezza provabile dimostrata da Dodis et al. si basa su un *seme casuale* indipendente dagli input, che assicura una distribuzione uniforme dell'entropia tra i pool.

Eliminando questa assunzione — come avviene nel paradigma pienamente *seedless* — la sicurezza non può più essere garantita: lo schema di assegnazione fisso adottato da Yarrow e Fortuna non è robusto contro gli attacchi *Premature-Next* senza introdurre restrizioni irrealistiche sull'avversario (ad esempio, tassi di input costanti). Sono state considerate diverse alternative *seedless*, come la scelta dinamica dei pool in funzione degli input, la suddivisione di ciascun contributo entropico in più parti o perfino la rimozione dei pool. Tuttavia, anche in questi scenari, la conclusione rimane invariata: **non esiste una costruzione *seedless* robusta anche in presenza di chiamate premature**, nemmeno nel caso in cui gli input entropici siano indipendenti.

Gli autori osservano inoltre che, per evitare risultati banali, è necessario porre alcune restrizioni sulle distribuzioni di input. In assenza di vincoli, infatti, nessuna costruzione può garantire sicurezza; tuttavia, anche nell'ipotesi più favorevole — in cui le sorgenti entropiche sono indipendenti e distribuite in modo uniforme — il risultato d'impossibilità continua a valere, rendendo ancora più solida la dimostrazione.

3.2 Impossibilità di sicurezza "*assoluta*" nei Seedless PRNG

Per analizzare formalmente la sicurezza dei generatori pseudocasuali *seedless* in presenza di chiamate premature, Coretti et al. [5] propongono un'estensione del classico gioco di robustezza dei PRNG, introducendo il modello *NROB* (*Non-Robustness with Premature Next*). L'obiettivo di questo modello è valutare la capacità del generatore di recuperare la sicurezza dopo una compromissione, anche quando l'avversario può invocare la procedura `next` prima che sia stata accumulata entropia sufficiente nello stato interno.

3.2.1 Il modello di sicurezza NROB

Il modello di robustezza classico (ROB — già visto nel Capitolo precedente) [9][11] valuta la sicurezza del PRNG assumendo che le chiamate a `next` avvengano solo

dopo che una quantità adeguata di entropia è stata accumulata. Nel modello NROB, tale vincolo viene rimosso: l'avversario può eseguire `next` in qualsiasi momento, anche quando lo stato è ancora "debole". Questo consente di modellare situazioni realistiche in cui l'entropia non è controllata o dove il sistema è costretto a produrre output troppo presto, come nelle fasi di avvio di un sistema operativo o di un protocollo crittografico.

Per semplificare l'analisi, gli autori assumono che i campioni entropici forniti dall'avversario siano indipendenti tra loro. Questa ipotesi non indebolisce il risultato, ma anzi lo **rafforza**: se l'impossibilità vale anche sotto input indipendenti, essa è valida a maggior ragione in scenari **più generali**. In termini formali, ogni input entropico X_i è estratto in modo indipendente da una distribuzione specificata dall'avversario, e il gioco di sicurezza considera campioni concreti $x_i \leftarrow X_i$ senza fornire ulteriori informazioni all'attaccante.

Descrizione del gioco NROB. Nel gioco di robustezza NROB, l'avversario A interagisce con tre oracoli che simulano il comportamento del PRNG:

- `get-next`, che richiama la funzione `next` e restituisce l'output pseudocasuale corrente;
- `next-ror`, che genera una sfida *real-or-random* in cui l'avversario deve distinguere se l'output proviene dal PRNG reale o da un generatore ideale;
- `get-state`, che restituisce lo stato interno del PRNG, simulando un evento di compromissione.

Durante il gioco, il sistema mantiene un contatore di entropia c, che rappresenta la quantità di entropia accumulata dopo l'ultima compromissione. Al raggiungimento di una soglia γ^*, il PRNG dovrebbe essere in grado di recuperare la sicurezza (*recovery*). Tuttavia, il modello NROB introduce un parametro aggiuntivo, β, che definisce il margine temporale concesso per il recupero: se l'entropia γ^* viene raggiunta dopo T^* passi, la sicurezza deve essere ristabilita entro βT^*. In altre parole, il parametro β controlla quanto ritardo è tollerato nella fase di recupero.

A questo punto, introduciamo i **criteri di sicurezza** nel contesto del modello NROB, chiarendo chi rappresenta il ruolo di attaccante e la misura del suo vantaggio. Un attaccante—(q, τ) è un avversario che può fornire al massimo q input al PRNG e che opera in tempo τ. La sua *advantage*, indicata come $\mathrm{Adv}^{\mathrm{NROB}}_{\mathrm{PRNG}}(A)$, misura la probabilità di distinguere il PRNG reale da un generatore ideale. Un PRNG è detto $(\gamma^*, \beta, q, \varepsilon, \tau)$-*sicuro* se, per ogni attaccante efficiente (q, τ), la probabilità di successo soddisfa $\mathrm{Adv}^{\mathrm{NROB}}_{\mathrm{PRNG}}(A) \leq \varepsilon$.

Confronto con il modello ROB. La differenza principale rispetto al modello ROB risiede nel trattamento del tempo di recupero. In ROB, il recupero è immediato ($\beta = 1$): appena l'entropia accumulata supera γ^*, il PRNG deve tornare sicuro. In NROB, invece, è consentito un ritardo proporzionale al fattore β. Questa *relaxation* consente di modellare comportamenti più realistici, specialmente nei PRNG che accumulano entropia in modo asincrono o dipendente da eventi esterni.

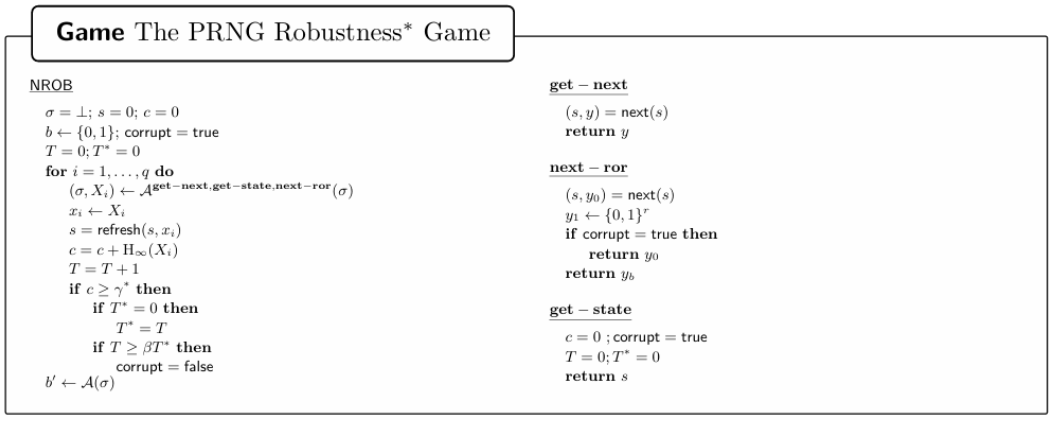


Figura 3.1. Il gioco di robustezza con chiamate premature *NROB(δ*,β,q)* introdotto in [5]. L'avversario interagisce con tre oracoli (`get-next`, `next-ror`, `get-state`) che simulano rispettivamente la generazione di output, la sfida *real-or-random* e la compromissione dello stato interno del PRNG.

3.2.2 Risultato di impossibilità

L'idea di fondo è che un avversario possa costruire una sequenza di input tale che ogni gruppo di n campioni contenga, nel complesso, un solo bit effettivo di entropia. In tale scenario, una chiamata prematura alla funzione `next` può rivelare informazioni su questo singolo bit, compromettendo quindi la sicurezza del PRNG nel modello NROB.

Per modellare questo comportamento, il generatore è rappresentato dalla funzione

$$
\begin{aligned}
&f_{\text{PRNG}}(s, x_1 \| \ldots \| x_n) : \\
&\quad \textbf{for } i = 1 \textbf{ to } n \textbf{ do} \\
&\qquad s = \texttt{refresh}(s, x_i) \\
&\quad (s, y) = \texttt{next}(s) \\
&\quad \textbf{return } y
\end{aligned}
$$

che applica n passi di aggiornamento (`refresh`) seguiti da una chiamata a `next`. Gli autori introducono inoltre la notazione x_{-i} per indicare la concatenazione di tutti gli input eccetto il bit x_i, e $x_{-i,\chi}$ per la stessa sequenza in cui x_i è sostituito da $\chi \in \{0,1\}$.

Per dimostrare l'impossibilità, viene introdotto un **algoritmo randomizzato** denominato FIND^g, che interagisce con una funzione $g : \{0,1\}^n \to \{0,1\}$. Con alta probabilità (almeno $1 - 2^{-n}$), l'algoritmo restituisce una delle seguenti situazioni:

1. la funzione g è sbilanciata, ossia $\Pr[g(U_n) = z] \geq 0.6$ per qualche $z \in \{0,1\}$;
2. esistono due stringhe $x, x_{-i,\chi}$ che differiscono in un solo bit e tali che $g(x) \neq g(x_{-i,\chi})$.

In altre parole, FIND^g trova un bias nell'output del generatore o due input quasi identici con output diversi, violando così la pseudocasualità del PRNG. L'algoritmo ha complessità $O(n^2)$, quindi è realistico per un avversario efficiente.

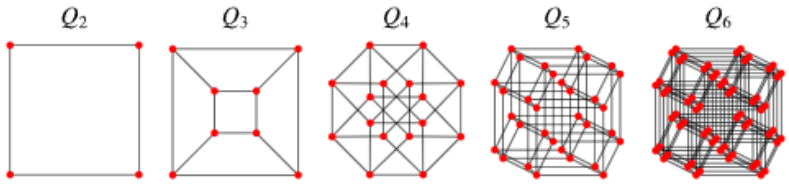


Figura 3.2. Rappresentazione di Grafo Ipercubico [34]

Analisi del primo ciclo (*First for Loop*). Per analizzare la prima parte dell'algoritmo FIND^g, si consideri il problema di determinare indici i e z tali che venga soddisfatta la seconda proprietà citata poco sopra. A tale scopo, gli autori fanno ricorso a risultati di teoria dei grafi, in particolare all'*ineguaglianza isoperimetrica* applicata a un grafo ipercubico (v. Figura 3.2) Nel nostro contesto, consideriamo l'ipercubo $Q_n = (V, E)$, dove ogni vertice di V rappresenta un vettore binario di lunghezza n, ossia $|V| = 2^n$. L'insieme E contiene tutti gli spigoli che connettono due vertici (u, v) aventi distanza di Hamming[1] pari a 1. In tal modo, ogni vertice ha grado n e il numero complessivo di spigoli risulta $|E| = n \cdot 2^{n-1}$. Siamo interessati agli spigoli che collegano due vertici u, v per i quali $g(u) \neq g(v)$, ossia coppie di input che producono output differenti. Sia $S \subseteq V$ un insieme di vertici di dimensione $k \leq 2^{n-1}$, e sia C il numero di spigoli "tagliati" che connettono S al suo complemento. L'ineguaglianza isoperimetrica per l'ipercubo fornisce il limite inferiore:

$$C \geq k \cdot (n - \log_2 k) \geq k,$$

questa disuguaglianza stabilisce che, man mano che il numero di vertici cresce, la quantità di connessioni tra regioni con output diversi cresce almeno linearmente in k. Dobbiamo ora stimare quanti vertici $u \in V$ soddisfano $g(u) = 0$ (o $g(u) = 1$). Se $0.4 \leq \mathbb{E}[g(U_n)] \leq 0.6$, significa che circa il 40% dei vertici assume valore 0 e un numero analogo assume valore 1. Ne segue che il numero di spigoli che connettono i due insiemi non è trascurabile, e la probabilità di selezionare casualmente uno spigolo "desiderato" è almeno:

$$\frac{k}{n \cdot 2^{n-1}} \geq \frac{0.4 \cdot 2^n}{n \cdot 2^{n-1}} = \frac{0.8}{n}.$$

In altri termini, la probabilità che uno spigolo scelto a caso sia quello corretto è $p \geq 0.8/n$, ovvero un singolo test casuale ha una probabilità non trascurabile di trovare una coppia di input con output differenti. Pertanto, l'algoritmo può semplicemente selezionare uno spigolo $e \in E$ in modo uniforme, verificarne il valore, e ripetere il test $O(n^2)$ volte.

Dopo un numero di prove dell'ordine di n^2, la probabilità di successo dell'algoritmo risulta almeno

$$\Pr[g(x_{-i,0}) \neq g(x_{-i,1})] > 1 - 2^{-n}.$$

[1]Nel contesto dell'analisi combinatoria dell'algoritmo FIND^g, la *distanza di Hamming* tra due stringhe binarie $u, v \in \{0,1\}^n$ misura il numero di bit che devono essere modificati per trasformare u in v. Due stringhe sono quindi adiacenti nell'ipercubo Q_n se la loro distanza è pari a 1, ossia se differiscono in un solo bit [35].

Questo risultato deriva dal fatto che la probabilità di fallimento dell'algoritmo è limitata da:

$$\left(1 - \frac{0.8}{n}\right)^{n^2} \leq e^{-0.8n} < 2^{-n},$$

il che garantisce che il successo avvenga con probabilità elevata, purché $0.4 \leq \mathbb{E}[g(U_n)] \leq 0.6$.

Algorithm FIND^g

for $i = 1$ **to** n^2:
  Pick an edge $(\mathbf{u}, \mathbf{v}) \in E$, uniformly at random.
  Use oracle access to g to compute $g(\mathbf{u})$ and $g(\mathbf{v})$.
  if $g(\mathbf{u}) \neq g(\mathbf{v})$ **then**
    Find i such that $u_i \neq v_i$.
    By definition, there exists a unique i that satisfies this condition.
    return (i, u_{-i})
    break
for $i = 1$ **to** $120 \cdot n$:
  $count = 0$
  Sample $\mathbf{x} \leftarrow \{0,1\}^n$
  Compute $count = count + g(\mathbf{x})$
if $count > n/2$ **then** $z = 1$
else $z = 0$
return $(0, z)$

Figura 3.3. Descrizione dell'algoritmo FIND^g tratto da [5]. Il primo ciclo ricerca due input che differiscono in un solo bit e producono output diversi; il secondo verifica la presenza di un bias statistico nella funzione g.

Analisi del secondo ciclo (*Second for Loop*). Se invece la funzione g risulta sbilanciata, ossia $\mathbb{E}[g(U_n)] < 0.4$ oppure $\mathbb{E}[g(U_n)] > 0.6$, la distribuzione degli output è inclinata verso uno dei due valori. Ad esempio, se $\mathbb{E}[g(U_n)] > 0.6$, significa che oltre il 60% degli input produce valore 1 (e meno del 40% valore 0). Applicando il limite di Chernoff, gli autori dimostrano che la probabilità di identificare correttamente tale bias è almeno $1 - 2^{-n}$, e quindi

$$\Pr[g(U_n) = z] \geq 0.6 \quad \text{con probabilità } 1 - 2^{-n}.$$

Con questo si completa la dimostrazione del funzionamento dell'algoritmo FIND^g, che individua o un bias statistico nella funzione g, oppure due input quasi identici che producono output diversi, contraddicendo la pseudocasualità del generatore.

In sintesi, tale analisi mostra che, in ogni caso, è possibile rilevare una deviazione dalla pseudocasualità di g, ponendo le basi per il teorema di impossibilità che segue.

Teorema 1. *Non esiste alcun PRNG completamente seedless che sia $(\gamma^*, \beta, q, 0.1, \tau)$-sicuro nel modello NROB quando $\gamma^*\beta < \sqrt{q}$ e $\tau \geq \Omega((t_{next} + t_{refresh})\, n^3)$, dove $n = \gamma^*\beta$ rappresenta il numero di iterazioni e $t_{next}, t_{refresh}$ i tempi richiesti per le rispettive procedure.*

Dimostrazione. La prova costruisce un avversario $\mathcal{A}$ capace di vincere il gioco NROB(γ^*, β) con probabilità costante, utilizzando come sottoprocedura l'algoritmo FIND^g definito poco prima. Si considera la funzione

$$g(x) = f(s, x_1 \| \dots \| x_n)$$

dove s è lo stato iniziale del PRNG e $n = \gamma^*\beta$. L'avversario conosce il primo stato s e applica FIND^g a questa funzione, ottenendo in uscita una coppia (i, z). L'attacco si divide quindi in due casi distinti.

Caso 1: $i = 0$**.** In questo caso, la funzione g risulta sbilanciata verso il valore z. L'avversario invoca l'oracolo `get-state` per azzerare i contatori di entropia e genera $n = \gamma^*\beta$ input pseudocasuali seguendo la distribuzione di $g(U_n)$, nota per essere biasata. Alla fine di questo processo, si ha $T^* = \gamma^*\beta$; l'attaccante invoca poi `next-ror` per ricevere la risposta di sfida. Se l'output coincide con z, conclude che la distribuzione è reale e vince il gioco.

Caso 2: $i \neq 0$**.** In questo caso esistono due stringhe di n bit che differiscono in una sola posizione e per le quali g produce output diversi. L'indice i identifica il bit divergente, mentre z rappresenta il valore atteso di output. L'avversario imposta il bit x_i nello stato, quindi procede con n iterazioni di aggiornamento tramite `refresh`. Durante questa fase, invoca una chiamata prematura a `get-next` e ottiene un output y. Conoscendo il valore z, $\mathcal{A}$ può calcolare le due possibili valutazioni di g e distinguere tra la distribuzione reale e quella casuale, ottenendo un vantaggio costante.

Poiché tale distinzione è possibile per qualunque coppia di parametri (γ^*, β) con $\gamma^*\beta < \sqrt{q}$, si conclude che **nessun** PRNG *seedless* può risultare **sicuro** nel modello NROB.

3.3 Analisi del Seedless Scheduler

Questa sezione può essere letta come un approfondimento conclusivo, in risposta ai problemi teorici discussi nel presente capitolo. Gli autori di [5] propongono infatti una ridefinizione del paradigma *seedless*, con l'obiettivo di individuare condizioni in cui esso possa risultare effettivamente robusto. Sebbene il risultato di impossibilità sia, in parte, di natura artificiale, esso apre la questione di come superare tale limite, anche assumendo un accumulo e un'estrazione ideali dell'entropia. A questo scopo, gli autori introducono lo **scheduler**[2], un modello teorico che descrive la distribuzione dell'entropia tra più *pool* e la sua gestione nel tempo.

3.3.1 Sintassi di uno Scheduler

Uno scheduler è definito come un algoritmo deterministico che produce una sequenza di coppie (in_i, out_i) per $i = 1, \ldots, q$, dove ciascun elemento specifica quale *pool* deve ricevere l'input entropico (in_i) e quale, eventualmente, deve svuotare il proprio contenuto (out_i). Il numero dei pool k non è critico ai fini della definizione: può essere visto come una funzione che, a ogni passo, determina se un pool è "*empty*" (cioè non deve ancora essere svuotato) oppure se è pronto a contribuire all'aggiornamento del registro principale. In termini intuitivi, uno scheduler SC con k pool e q

[2]Nel nostro contesto, il termine scheduler non va confuso con i key scheduler dei cifrari simmetrici (come AES o DES): qui rappresenta invece un "controllore deterministico" che regola il flusso e la distribuzione dell'entropia nei vari pool del generatore. In questo modello, il processo di accumulo e di estrazione è completamente astratto, e lo scheduler diventa il meccanismo centrale che coordina le fasi di aggiornamento del PRNG in modo indipendente dai dati effettivi in ingresso.

iterazioni definisce una **politica di aggiornamento** che distribuisce e rilascia entropia secondo pesi normalizzati $w_i \in [0, 1]$, tali che un pool si considera "pieno" quando ha accumulato peso 1. Questo modello consente di astrarre la dinamica reale dei generatori come Yarrow o Fortuna, concentrandosi solo sul *quando* l'entropia viene gestita, piuttosto che sul suo valore numerico.

3.3.2 Composizione con PRNG Seedless

Per studiare la sicurezza di uno scheduler, gli autori lo combinano con una famiglia di PRNG *seedless* non resistenti ad attacchi di tipo *Premature-Next*. Ogni generatore $G_i = (\mathsf{refresh}_i, \mathsf{next}_i)$ gestisce uno dei pool interni, mentre lo scheduler determina quale di essi deve ricevere il nuovo input entropico e quale deve rilasciare il proprio contenuto verso il registro principale.

Formalmente, la composizione di questi moduli definisce un PRNG complessivo $\mathcal{G}(\mathsf{SC}, \{G_i\}_{i=0}^{k-1}, G)$, in cui lo scheduler SC decide le chiamate a $\mathsf{refresh}$ e a next. Ai fini dell'analisi teorica, l'accumulo e l'estrazione di entropia sono considerati *ideali*: lo scheduler non misura la quantità reale di entropia, ma ne gestisce solo la tempistica.

Questa astrazione rende possibile isolare il ruolo del controllo temporale nella sicurezza dei PRNG seedless, senza dipendere da ipotesi specifiche sulle fonti entropiche o sulla loro distribuzione.

3.3.3 Sicurezza e limiti del modello di Scheduler

Come per i PRNG tradizionali, anche per gli scheduler è possibile introdurre una nozione di sicurezza parametrizzata da due valori, α e β, che rappresentano rispettivamente la quantità di entropia fresca iniettata nel sistema e l'intervallo di tempo entro il quale si richiede il recupero (*recovery*) dopo una compromissione. In modo intuitivo, si vuole garantire che, se un avversario fornisce almeno α unità di entropia entro un tempo $t \leq q/\beta$, il sistema torni sicuro entro $t + \beta t$.

Un (k, q)-*scheduler* è detto (α, β)-*general-secure* se, in ogni intervallo temporale $[t_0, t_0 + \beta t]$, esiste almeno un pool che soddisfa le seguenti condizioni:

1. il pool non è stato svuotato prima del tempo $\hat{T}$;
2. il pool viene svuotato esattamente al tempo $\hat{T}$;
3. il pool ha accumulato entropia sufficiente, cioè $\sum_{i=t_0}^{\hat{T}} w_i \geq 1$.

Queste tre condizioni esprimono la garanzia che lo scheduler riesca a sincronizzare correttamente le fasi di accumulo e rilascio dell'entropia, fornendo così un modello astratto di recupero post-compromissione.

Risultato di impossibilità. Secondo il **Teorema 8** del paper (cfr. Sezione 4.4 [5]) si dimostra che, anche per questa definizione di sicurezza, esiste un limite teorico intrinseco. In particolare, per ogni $k \in \mathbb{N}$, esiste un valore di q tale che nessuno scheduler (k, q) risulta (α, β)-sicuro. In altre parole, scegliendo opportunamente il numero di iterazioni q, è possibile costruire un avversario che impedisce al sistema di recuperare la sicurezza per qualunque coppia (α, β).

Il teorema formale stabilisce che:

$$q^* = \alpha^2\beta^2 \quad \Rightarrow \quad (k, q)\text{-scheduler non è } (\alpha, \beta)\text{-sicuro per ogni } q \geq q^*.$$

La dimostrazione si basa sulla costruzione di due scenari alternativi:

- nel primo caso, nessun pool viene svuotato durante un intervallo $\alpha\beta$; il sistema rimane quindi bloccato, incapace di recuperare la sicurezza;
- nel secondo caso, almeno un pool viene svuotato troppo presto, prima che abbia accumulato entropia sufficiente, vanificando di fatto tutto l'apporto informativo.

In entrambi i casi, l'avversario riesce a forzare una condizione di *non-recovery*, dimostrando che nessuno scheduler deterministico può garantire la sicurezza per ogni possibile sequenza di input. Con questo risultato si completa l'analisi teorica del modello di scheduler, dimostrando che anche in presenza di più pool e di un controllo temporale ottimale, il paradigma *seedless* **non può ottenere sicurezza assoluta** contro gli attacchi di tipo *Premature-Next*.

3.4 Scheduler Sicuri al Riavvio e Modelli Ripetitivi

In questa sezione discutiamo di modelli più restrittivi ma realistici, che permettono di formulare risultati positivi in contesti pratici, ossia varie estensioni (*relaxations*) del modello definito nella sezione precedente. Le analisi che seguono esaminano tre varianti del Seedless Scheduler: la *sicurezza al riavvio*, gli *scheduler ripetitivi* e la loro estensione *liscia* (*smooth*), che rappresenta il limite di efficienza raggiungibile da tali sistemi. *Per una trattazione completa dei risultati presentati in questa sezione, si rimanda alle Sezioni 5–6 di [5].*

3.4.1 Modello di Sicurezza al Riavvio (Reboot Security)

Il primo "rilassamento" riguarda lo scenario in cui il sistema è appena riavviato, cioè $t_0 = 0$. In questa fase il PRNG non dispone ancora di entropia accumulata e deve ripristinare gradualmente la sicurezza a partire da uno stato nullo. Si introduce quindi il concetto di **reboot security**, che formalizza la capacità dello scheduler di recuperare da una compromissione immediata dopo l'avvio.

Un (k, q)-*scheduler* è detto (α, β)-*reboot-secure* se, per ogni sequenza di pesi $w_1, \ldots, w_q \in [0, 1]$ che rappresenta la quantità di entropia fornita dall'ambiente, esiste un intervallo temporale di lunghezza al più βt entro cui almeno un pool accumula entropia totale $\sum_i w_i \geq \alpha$. In tal caso, lo scheduler è in grado di *recuperare* entro tempo $t + \beta t$.

Rispetto al modello generale, l'ipotesi $t_0 = 0$ semplifica l'analisi poiché l'avversario non può scegliere arbitrariamente il punto di compromissione. Secondo il **Teorema 10** (cfr. Sezione 5.2 [5]), per garantire questa proprietà è necessario che la quantità di entropia iniettata α cresca almeno in modo logaritmico con il numero di iterazioni q, ossia $\alpha = \Omega(\log_\beta q)$, mentre β rimane un fattore costante. L'intuizione è che, in ogni intervallo di ampiezza geometrica β^i, deve avvenire almeno

uno svuotamento capace di rilasciare un'unità di entropia; in caso contrario, il sistema non potrebbe mai ristabilire la sicurezza.

Un esempio concreto di scheduler reboot-secure è quello utilizzato nel generatore di Windows 10, in cui un singolo pool di entropia viene svuotato a passi crescenti β^i con $\beta \approx 3$. Questo schema è noto come **Root RNG** e rappresenta una realizzazione pratica del modello: un unico pool, inizialmente vuoto, che si svuota con frequenza esponenzialmente decrescente, permettendo una ricostruzione graduale ma sicura dello stato del PRNG.

3.4.2 Scheduler Ripetitivi e Round-Robin

Rispetto al modello di sicurezza al riavvio, il caso ripetitivo rappresenta un modello più generale, in cui anche il momento iniziale della compromissione t_0 può essere scelto dall'avversario. Ciò rende il modello più realistico e, al tempo stesso, più difficile da analizzare. Il risultato di impossibilità precedente impone pertanto di introdurre ulteriori vincoli strutturali sullo scheduler.

Si considerano quindi i *repeat-secure schedulers*, nei quali la distribuzione dell'entropia segue una politica regolare e ripetitiva su più pool. In un'implementazione tipica, lo scheduler adotta una strategia di tipo *round-robin*, distribuendo ciclicamente gli input entropici tra k pool distinti. Ogni round corrisponde a un intervallo temporale di lunghezza β^i e, al termine di ciascun ciclo, uno dei pool viene svuotato e combinato nel registro principale. Questo meccanismo riproduce il comportamento di sistemi come `/dev/random` o `Fortuna`, che alternano fasi di accumulo e rilascio in modo regolare per mantenere un livello di entropia costante.

Dal punto di vista teorico, la sicurezza ripetitiva è definita rispetto a *sequenze k-ripetitive* di pesi w_i, ossia pattern periodici che modellano un tasso di entropia che si ripete nel tempo. Per gestire le fluttuazioni locali nella qualità dell'entropia, viene introdotta la nozione di *k-flattening*, che sostituisce in ogni round-robin l'entropia effettiva con il minimo tra i k valori contribuiti dai singoli pool. In questo modo, si assegna all'intero round l'entropia del pool più debole, garantendo che la misura di sicurezza non sovrastimi mai la qualità della sorgente.

3.4.3 Scheduler Lisci e Analisi Asintotica

L'ultima estensione del modello introduce i cosiddetti *smooth schedulers*, che generalizzano lo schema round-robin verso un comportamento ancora più regolare. Uno scheduler (k, q)-smooth è parametrizzato da due interi b e k, con $q = b^k$. L'idea è di combinare la periodicità del round-robin con una distribuzione di svuotamenti scalata gerarchicamente in base ai parametri (b, k), in modo da evitare picchi improvvisi di entropia o lunghi intervalli di inattività.

In configurazioni pratiche si assumono valori come $b = 2$ o 3 e $k = O(\log q)$, per i quali il fattore di recupero β tende a diventare costante. Secondo il **Teorema 17** (cfr. Sezione 6.3 [5]), tali scheduler sono (α, β, k)-repeat-secure con parametri esatti

$$\alpha = 3k - 2 \quad e \quad \beta = 2b(1 + k/\alpha).$$

In configurazioni pratiche, ponendo $q = b^k$ con $b \in \{2, 3\}$ e $k = O(\log q)$, si ottiene quindi

$$\alpha = O(\log q), \qquad \beta = O(1),$$

ossia una crescita logaritmica dell'entropia richiesta e un fattore di recupero pressoché costante.

Ciò implica che, aumentando il numero di pool, il tempo di recupero del sistema rimane stabile senza richiedere maggiore entropia complessiva. In termini operativi, la sicurezza diventa quasi indipendente dal numero di iterazioni, rendendo il modello adatto a PRNG multi-pool di lungo ciclo, come `Fortuna` o i generatori basati su `SHA-3`.

Conclusioni

Nel corso di questo lavoro si è analizzato il problema della generazione pseudo-casuale in ambienti privi di seme affidabile, con particolare attenzione ai modelli e ai risultati più recenti relativi ai PRNG *seedless*. A partire dal quadro teorico introdotto in CRYPTO/2019 e successivamente esteso nel contesto in ITC/2022, la tesi ha mostrato come l'eliminazione del seed non sia soltanto una scelta pragmatica che potrebbe esser adottata dai sistemi reali, ma anche una direzione teorica ricca di implicazioni.

I risultati esistenti suggeriscono una situazione duale: da un lato, costruzioni seedless sono possibili e garantiscono robustezza sia a livello computazionale che informazionale in modelli ideali o in scenari con input indipendenti; dall'altro, l'estensione di tali garanzie a modelli più realistici incontra limiti strutturali, evidenziato dall'impossibilità di ottenere sicurezza concreta contro attacchi di tipo *premature-next* in generale. Questo fenomeno sottolinea come, nell'assenza di un seme, l'interazione fra accumulo di entropia, compromissione dello stato e tempistiche delle operazioni *next* di recupero diventi un punto critico del design.

Le analisi relative agli schedulers mostrano invece un'area promettente. Euristiche come il *root pool* o il *round-robin* scheduler risultano efficaci in contesti specifici, purché accompagnati da ipotesi naturali sull'andamento dell'entropia (ad esempio, da *k-smooth entropy*). Tali modelli rappresentano oggi una delle principali frontiere teoriche per aggirare le impossibilità note, trasformando intuizioni matematiche e ingegneristiche in garanzie formali.

Un insieme interessante di prospettive emergenti riguarda il modo in cui idee proprie del paradigma seedless compaiono (anche implicitamente) in implementazioni reali. Il *Linux–Deterministic Random Bit Generator* (Linux–DRBG), ad esempio, presenta proprietà affini ai PRNG seedless grazie all'uso di più sorgenti di entropia e a un meccanismo di accumulo robusto. Analogamente, nei *Quantum Random Number Generators* (QRNG) estrattori online basati su forward block sources operano senza seed e gestiscono flussi entropici elevati, riproponendo concetti vicini a quelli degli estrattori seedless.

Questi sviluppi indicano una direzione di convergenza interessante: contesti diversi – dalla progettazione dei kernel, alla crittografia teorica, fino ai sistemi quantistici e tanti altri – stanno adottando modelli in cui la tradizionale distinzione tra seed, entropia e stato interno viene progressivamente ripensata. Pur mancando ancora una teoria unificata, l'impressione complessiva è che il paradigma seedless stia iniziando a influenzare in modo trasversale strumenti, modelli e implementazioni, delineando uno spazio di ricerca ampio e ancora in gran parte inesplorato.

Bibliografia

[1] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology — CRYPTO'96*. Springer, 1996.

[2] Mihir Bellare and Phillip Rogaway. The security of pseudorandom functions. *Journal of Cryptology*, 1993.

[3] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT Hash Workshop*, 2007.

[4] Katrina Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016.

[5] Sandro Coretti, Yevgeniy Dodis, Harish Karthikeyan, Noah Stephens-Davidowitz, and Stefano Tessaro. On seedless PRNGs and premature next. In *Proceedings of the Conference on Information-Theoretic Cryptography (ITC)*, 2022. Available at Cryptology ePrint Archive, Paper 2022/558.

[6] Joan Daemen and Gilles Van Assche. The keccak reference. *Submission to NIST (Round 3)*, 2013.

[7] Yevgeniy Dodis, Siyao Guo, Jonathan Katz, Leonid Reyzin, and Adam Smith. Seedless fruit is the sweetest: Random number generation without the seed. Cryptology ePrint Archive, Paper 2019/198, 2019.

[8] Yevgeniy Dodis, Shachar Lovett, Steven Vadhan, and David Zuckerman. Privacy amplification and nonmalleable extractors via character sums. In *FOCS 2008*. IEEE Computer Society, 2008.

[9] Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergnaud, and Daniel Wichs. Security analysis of pseudorandom number generators with input: Robustness and forward security. *Algorithmica*, 2017.

[10] Yevgeniy Dodis and Prashant Puniya. On the relation between the ideal cipher and the random oracle models. Technical report, Cryptology ePrint Archive, Report 2006/276, 2006. Studio formale dell'Ideal Cipher Model (E-model) e della sua relazione con il Random Oracle Model.

[11] Yevgeniy Dodis, Leonid Reyzin, and Stefano Tessaro. Robust pseudorandom generators and separating pseudorandomness from entropy. In *Advances in Cryptology – EUROCRYPT 2018*, Lecture Notes in Computer Science, Cham, 2018. Springer.

[12] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. Wiley, 2003. Includes the Fortuna pseudorandom number generator.

[13] R. A. Fisher and F. Yates. *Statistical Tables for Biological, Agricultural and Medical Research*. Oliver and Boyd, 1938.

[14] Danny Harnik and Moni Naor. On everlasting security in the hybrid bounded-storage model. In *Advances in Cryptology - CRYPTO 2006*, Lecture Notes in Computer Science. Springer, 2006.

[15] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Yarrow-160: Design and analysis of the Yarrow cryptographic pseudorandom number generator. In *Fast Software Encryption (FSE)*, 1998.

[16] M. G. Kendall and B. Babington-Smith. Randomness and random sampling numbers. *Journal of the Royal Statistical Society. Series B (Methodological)*, (2), 1939.

[17] Donald E. Knuth. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*. Addison-Wesley, 3rd edition, 1997.

[18] Hugo Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. In *Advances in Cryptology - CRYPTO 2010*, Lecture Notes in Computer Science. Springer, 2010.

[19] Pierre L'Ecuyer. Uniform random number generation. *Annals of Operations Research*, (1), 1994.

[20] Pierre L'Ecuyer. History of uniform random number generation. In *2017 Winter Simulation Conference (WSC)*, Las Vegas, NV, USA, 2017. IEEE.

[21] D. H. Lehmer. Mathematical methods in large-scale computing units. *Proceedings of the Second Symposium on Large-Scale Digital Calculating Machinery*, 1949.

[22] George Marsaglia. Xorshift rngs. *Journal of Statistical Software*, 2003.

[23] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 1998.

[24] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[25] National Institute of Standards and Technology. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical Report Special Publication 800-22 Rev. 1a, NIST, 2010.

[26] National Institute of Standards and Technology. Recommendation for random number generation using deterministic random bit generators. Technical Report Special Publication 800-90A Rev. 1, NIST, 2015.

[27] Noam Nisan and David Zuckerman. Randomness is linear in space. In *Journal of Computer and System Sciences*. Academic Press, 1996.

[28] Melissa E. O'Neill. Pcg: A family of simple fast space-efficient statistically good algorithms for random number generation. In *Proceedings of the ACM SIGPLAN International Symposium on Programming Language Design and Implementation (PLDI)*. ACM, 2014.

[29] RAND Corporation. *A Million Random Digits with 100,000 Normal Deviates*. The RAND Corporation, Glencoe, IL, 1955.

[30] Bruce Schneier and Niels Ferguson. *Practical Cryptography*. Wiley, 2003.

[31] Bruce Schneier, Niels Ferguson, John Kelsey, Chris Hall, and David Wagner. Yarrow cryptographic pseudorandom number generator. Technical report, Counterpane Internet Security, 1999.

[32] L. H. C. Tippett. *Random Sampling Numbers*. Tracts for Computers, Cambridge University Press, 1927.

[33] John von Neumann. Various techniques used in connection with random digits. *Applied Math Series*, 1951. Original manuscript 1949.

[34] Eric W. Weisstein. Hypercube graph. `https://mathworld.wolfram.com/HypercubeGraph.html`, 2025. Wolfram MathWorld, accessed on November 6, 2025.

[35] Wikipedia contributors. Hamming distance. `https://en.wikipedia.org/wiki/Hamming_distance`, 2025. Accessed on November 6, 2025.